NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

DESIGN OF A PHASE SAMPLED INTERFEROMETRY ANTENNA USING THE ROBUST SYMMETRICAL NUMBER SYSTEM

by

Nathan S. York

December 2000

Thesis Advisor: Phillip E. Pace
Co-advisor: D. Scott Davis

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
Design of a Phase Sampled Interferometry Antenna Using the Robust Symmetrical Number System

5. FUNDING NUMBERS

6. AUTHOR(S)
York, Nathan S.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Center for Joint Services Electronic Warfare
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(maximum 200 words)*

This research has examined the benefits of using the Robust Symmetrical Number System (RSNS) to resolve ambiguities in phase sampling interferometry. A compact, high resolution direction finding antenna architecture based on the RSNS was developed to demonstrate experimentally the elimination of phase errors using a minimum amount of hardware. Previous work has determined that phase errors in the system will degrade the system performance. Several improvements were made to the original RSNS prototype antenna to provide enhanced performance. Adding isolators and supplementing the ground plane with copper tape (between the antenna elements), a reduction in the mutual coupling effects was accomplished. Mounting the microwave components on a brass plate also reduced errors contributed by vibrations and temperature. Tailor cutting all semi-rigid coaxial lines also helped reduce the number of connectors required to assemble the microwave circuit, also a source of phase errors. Matching the front-end amplifiers in each amplification stage rather than matching the characteristics of two cascaded amplifiers in each signal line has reduced relative phase errors between channels as well as matching the power outputs of the amplifiers. Two printed circuit boards were designed and built for the RSNS signal processor. The printed circuit boards provide a decrease in the electrical noise floor over the original design (assembled on breadboards). The new design has reduced the phase errors that were present in the first prototype system. The RSNS signal processing technique is able to provide a high-resolution phase sampled direction finding capability with an angular resolution of 1.9 degrees by using only three receiving elements (two interferometers).

14. SUBJECT TERMS
Robust symmetrical number system; Optimum symmetrical number system; Phase sampling interferometry; Direction finding, Ambiguity resolution.

15. NUMBER OF PAGES
144

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

DESIGN OF A PHASE SAMPLED INTERFEROMETRY ANTENNA USING THE ROBUST SYMMETRICAL NUMBER SYSTEM

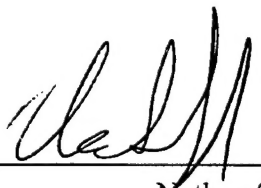
Nathan S. York
Lieutenant, United States Navy
B.S., Worcester Polytechnic Institute, 1994

Submitted in partial fulfillment of the
Requirements for the degree of

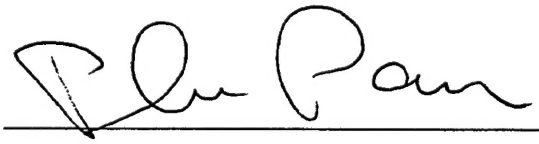
MASTER OF SCIENCE IN APPLIED PHYSICS

from the

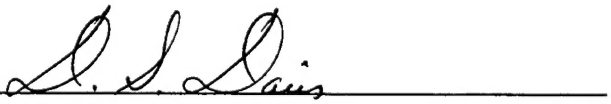
NAVAL POSTGRADUATE SCHOOL
December 2000

Author: 

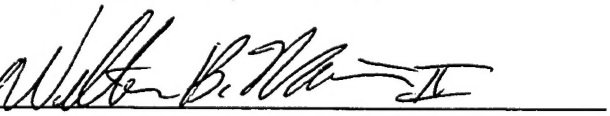
Nathan S. York

Approved by: 

Phillip E. Pace, Thesis Advisor



D. Scott Davis, Co-advisor



William B. Maier II, Chair
Department of Physics

ABSTRACT

This research has examined the benefits of using the Robust Symmetrical Number System (RSNS) to resolve ambiguities in phase sampling interferometry. A compact, high resolution direction finding antenna architecture based on the RSNS was developed to demonstrate experimentally the elimination of phase errors using a minimum amount of hardware. Previous work has determined that phase errors in the system will degrade the system performance. Several improvements were made to the original RSNS prototype antenna to provide enhanced performance. Adding isolators and supplementing the ground plane with copper tape (between the antenna elements), a reduction in the mutual coupling effects was accomplished. Mounting the microwave components on a brass plate also reduced errors contributed by vibrations and temperature. Tailor cutting all semi-rigid coaxial lines also helped reduce the number of connectors required to assemble the microwave circuit, also a source of phase errors. Matching the front-end amplifiers in each amplification stage rather than matching the characteristics of two cascaded amplifiers in each signal line has reduced relative phase errors between channels as well as matching the power outputs of the amplifiers. Two printed circuit boards were designed and built for the RSNS signal processor. The printed circuit boards provide a decrease in the electrical noise floor over the original design (assembled on breadboards). The new design has reduced the phase errors that were present in the first prototype system. The RSNS signal processing technique is able to provide a high-resolution phase sampled direction finding capability with an angular resolution of 1.9 degrees by using only three receiving elements (two interferometers).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

ADC	Analog to Digital Converter
AOA	Angle of Arrival
BW	Bin Width
d_i	Element Spacing for Element i
d'	Element Spacing for Scaled Array
∂	Forced Phase Difference
CAM	Computer Aided Manufacturing
DC	Direct Current
DF	Direction Finding
DIO	Digital Input-Output card
EEPROM	Electrically Erasable Programmable Read Only Memory
f	Frequency
FM	Frequency Modulated
ϕ	Broadside Phase Difference
g	Discrete state of RSNS vector
GPIB	General Purpose Interface Bus (synonymous with HPIB)
HPIB	Hewlett-Packard Interface Bus (synonymous with GPIB)
I_j	Current through Voltage Divider Resistance of *j*-th Comparator
IF	Intermediate Frequency
k	Wave Number
$\hat{k}$	Bin Number
L	Comparator Output Voltage
LNA	Low Noise Amplifier
LPF	Low Pass Filter
λ	Wavelength
$M, \hat{M}$	Dynamic Range for OSNS, RSNS
m, m_i	Modulus

N	Number of Channels
n_i	Number of Folds
η	Normalized Phase Detector Output
OSNS	Optimum Symmetrical Number System
PF_{RSNS}	Fundamental Period
PRP	Pairwise Relatively Prime
PSI	Phase Sampled Interferometry
PCB	Printed Circuit Board
ϕ	Wave Function of Free Space Wave
R_j	Resistance
r_k	Resolution of Bin k
RDF	Radio Direction Finding
RSNS	Robust Symmetrical Number System
s_i	Channel Sequence Shift Value
SNS	Symmetrical Number System
θ	Angle of Arrival
$\hat{\theta}$	Reported Angle of Arrival (Output of SNS system)
θ'	Angle of Arrival for Scaled Array
θ_{mm}	Maximum Mapable Aperture
V	Voltage
V_{j,m_i}	Threshold Voltage for j-th Comparator and i-th Modulus
V_{out}	Output of Amplifier Stage
V_{thresh}	Comparator Threshold Voltage
VHDL	VHSIC Hardware Description Language
VNA	Vector Network Analyzer
ξ	Scale Factor
ψ	Wave function at a receiving element

ACKNOWLEDGEMENTS

First of all, I would like to thank Professor Phillip Pace for allowing me to continue work on this project. His guidance, enthusiasm and patience made this an enjoyable experience. Next I would like to thank Professor Scott Davis, for his careful review of this thesis and thoughtful comments. This thesis would not have progressed if it were not for the guidance of Professor David Jenn, the Microwave Laboratory Director. I need to also thank Don Snyder for his help with the printed circuit board manufacturing and Bob Vitale for his efforts to help me understand the test equipment in the anechoic chamber. To David Styer, thank you for your hard work laying the foundation on which this thesis is built and to David Wickersham who built the first RSNS prototype antenna and introduced me to this thesis. Finally, I need to thank my wife, Vickie, and my children, Garrett and Margaret, for their support and understanding through this whole period.

This research was supported in part by the Naval Postgraduate School Center for Joint Services Electronic Warfare.

I. INTRODUCTION

A. DIRECTION FINDING ANTENNAS

Direction Finding (DF) antennas are used by many people. Law enforcement, military, wildlife managers and telecommunications personnel use DF systems for locating and tracking different types of emitters. A wide variety of techniques are used to accomplish the DF task. Spinning DF, time difference of arrival (TDOA), and phase sampled interferometry are just a few examples that are commonly used. All these systems are passive; meaning they do not generate their own signal to locate a target. Therefore most of these systems will only return a line of bearing to a transmission.

Phase sampled interferometry is a very attractive technique since a direction of arrival can be obtained by the phase information stored in a plane wave. Amplitude information is not required. Within the phase sampled interferometry field of direction finding, several new techniques have been investigated.

The *ESPIRIT-based two-dimensional arrival estimation* scheme achieves aperture extension (interferometry baseline extension) using a sparse uniform rectangular array of electromagnetic vector sensors spaced much farther apart than a half-wavelength [1]. An electromagnetic vector sensor is composed of six spatially co-located, orthogonally orientated, diversely polarized antennas, distinctly measuring all six electromagnetic-field components of an incident multisource wavefield. The direction of arrival from each incident source is estimated from the source's electromagnetic-field vector component and serves as a course reference to remove the cyclic phase ambiguities in ESPIRIT's eigenvalues when the intervector sensor spacing exceeds a half-wavelength.

A *polynomial rooting approach to super-resolution array design* is concerned with the design of an array that satisfies prespecified performance levels, such as detection-resolution thresholds and Cramér-Rao bounds on error variance [2]. The sensor placement problem is formulated in the framework of subspace-based DF techniques and a novel polynomial rooting approach to the design problem, based on the new concept of

the "sensor locator polynomial." This polynomial is constructed using prespecified performance levels, and its roots yield the sensor locations of the desired array. The distinguishing feature of this technique is that it hinges on the properties of the array manifold.

The *extended phase interferometry* technique [3] incorporates both calibrated phase and amplitude response data from the antenna arrays. This technique appropriately weights the square of the baseline phase differences with the antenna gains. The incorporation of amplitude data provides significant performance improvement over phase-only interferometry; however, it requires a modest increase in computational complexity.

This thesis research improves on a phase sampling interferometer approach that can be easily incorporated into the established techniques to provide a high-resolution, small baseline array with fewer number of phase sampling comparators [4]. The approach is based on preprocessing the received signal using the robust symmetrical number system (RSNS). The RSNS preprocessing is used to decompose the spatial filtering operation into a number of parallel sub-operations (moduli) that are of smaller computational complexity. Each sub-operation is a separately configured interferometer that symmetrically folds the detected phase difference with folding period equal to $2Nm_i$ where N is the number of interferometers that are used within the linear array. A small comparator ladder mid-level quantizes each folded phase response. Consequently, each sub-operation only requires a precision in accordance with that modulus. A much higher DF spatial resolution is achieved after the N different moduli are used and the results of these low precision sub-operations are recombined. By incorporating RSNS preprocessing concept, the field of view of a specific configuration of interferometers and phase sampling comparator arrangements can be analyzed exactly. Experimental results for an improved 6-bit RSNS array are presented.

B. PRINCIPAL CONTRIBUTIONS

The focus of this thesis research has been to improve the design and performance of a previous RSNS direction finding array. Microwave component performance was re-evaluated and new features were introduced. A new RSNS DF system was fabricated and tested.

Initial design efforts focused on the microwave receiver. The previous prototype's performance was marginal. Its design injected distortions into the folding waveforms, causing angle of arrival reporting errors. The existing component performances were re-examined and a new design was constructed.

The previous RSNS digital processing circuit was designed and constructed without its performance ever being validated. Computer simulations of the new circuit were conducted to evaluate the current design. During the simulations, it was discovered that by adding a timing circuit to clock in comparator outputs to the RSNS encoders, the performance of the system could be improved.

The new RSNS processing circuit was laid out for construction on a printed circuit board. The circuit board was milled on campus in the Physics department electronics workshop.

The improved prototype was tested. Data collection was conducted by collecting mixer output voltages and RSNS processor outputs. The measured patterns were used to study the effects of the new design features on mutual coupling, amplifier harmonics, temperature, vibrations and power levels.

C. THESIS OUTLINE

This thesis presents the theoretical background and design equations needed to construct a phase sampled interferometer, as well as previous test arrays that have been built and tested. A new array was designed to eliminate phase errors that were present in the preceding systems.

Chapter II is a review of phase sampled interferometry and the prototype antenna design elements. Chapter III introduces the Optimum Symmetrical Number System (OSNS), taken from the work of Thomas Hatziathanasiou [4]. It presents the design of a DF system based on the OSNS. Chapter IV, taken from the work of David Wickersham [5], gives an overview of previous work done with the Robust Symmetrical Numbering Systems and its application in DF antenna systems. Chapter V presents my design data for the antenna and RF receiver components. Chapter VI is a detailed description of my RSNS signal processing design. Chapter VII is an explanation of the testing procedures for the antenna and a presentation of the experimental results. Chapter VIII gives some concluding remarks and summarizes the next step that is needed.

II. PHASE SAMPLED INTEFEROMETRY

A two-element linear interferometer is shown in Figure 2.1. The two antenna elements are spaced a distance *d* apart and the incident plane wave arrives with bearing angle θ_B[5]. In this phase monopulse configuration the angle θ_B is measured from the perpendicular to the baseline axis and can take on values $\pi/2 > \theta_B > -\pi/2$. The phase difference between the two elements is

$$\Delta\psi = \psi_1 - \psi_2 = \frac{2\pi}{\lambda} d\sin(\theta_B) \tag{2.1}$$

where d is the element spacing and λ is the wavelength, and is a function of the incidence angle of the wave.

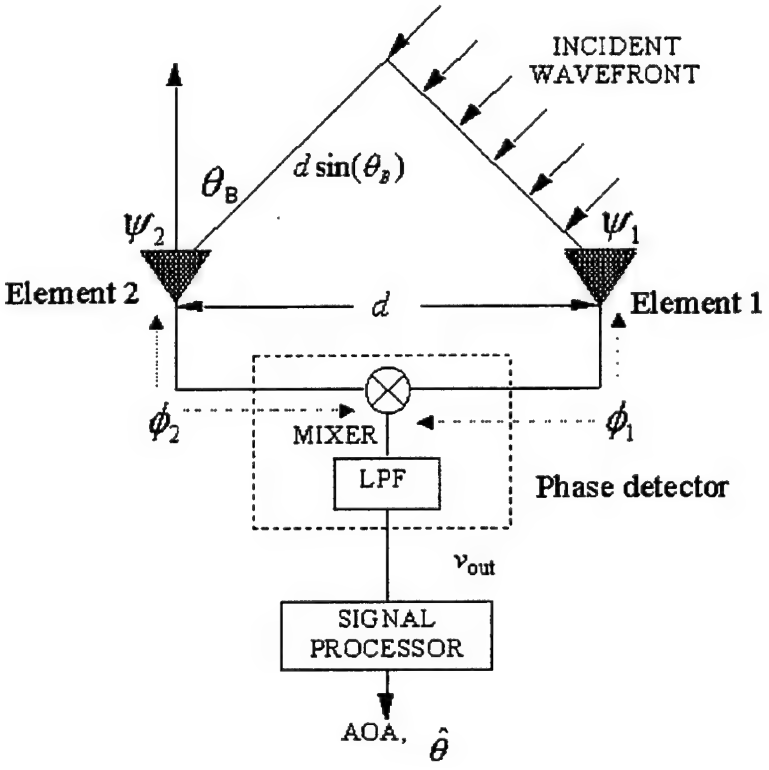


Figure 2.1: Two Element Interferometry.

The signals received by the antenna elements are mixed (multiplied together and lowpass filtered) resulting in an output signal whose frequency is the difference of the two input signal frequencies. Because the signals have the same frequency, the mixer output voltage is a value proportional to the difference between signal phases ψ_1 and ψ_2. This difference is not purely $\Delta\psi$ because of the time delays due to the different transmission line lengths, ϕ_i, from each antenna element. However, these line lengths are known and can be compensated for in the angle estimate. Let the signals from the two antenna elements be

$$v_1(t) = V\cos[2\pi ft + \phi_1(t)] \tag{2.2}$$

and

$$v_2(t) = V\cos[2\pi ft + \phi_2(t)], \tag{2.3}$$

where V is the maximum value of the voltage at the antenna elements, t is time and f is frequency. Let ψ_0 be the phase difference between the transmission lines to the two elements. The lowpass mixer output voltage is

$$v_{out} = \frac{V^2}{2}\cos(\Delta\phi) = \frac{V^2}{2}\cos\left(\frac{2\pi d}{\lambda}\sin(\theta_B) + \psi_0\right), \tag{2.4}$$

which contains the plane wave angle of arrival (AOA) information. For values of $d = \lambda/2$ and $\psi_0 = 0$, $\Delta\phi = \pi\sin(\theta_B)$. As the AOA θ_B varies from $-\pi/2$ to $\pi/2$, the phase difference $\Delta\phi$ varies from $-\pi$ to π as shown in Figure 2.2. The output voltage from the phase detector is also a function of the phase difference and is a symmetrical folding periodic waveform. Together these relationships give the phase detector output voltage as a symmetrical function of the AOA as shown in Figure 2.3 for $d = \lambda/2$ [6].

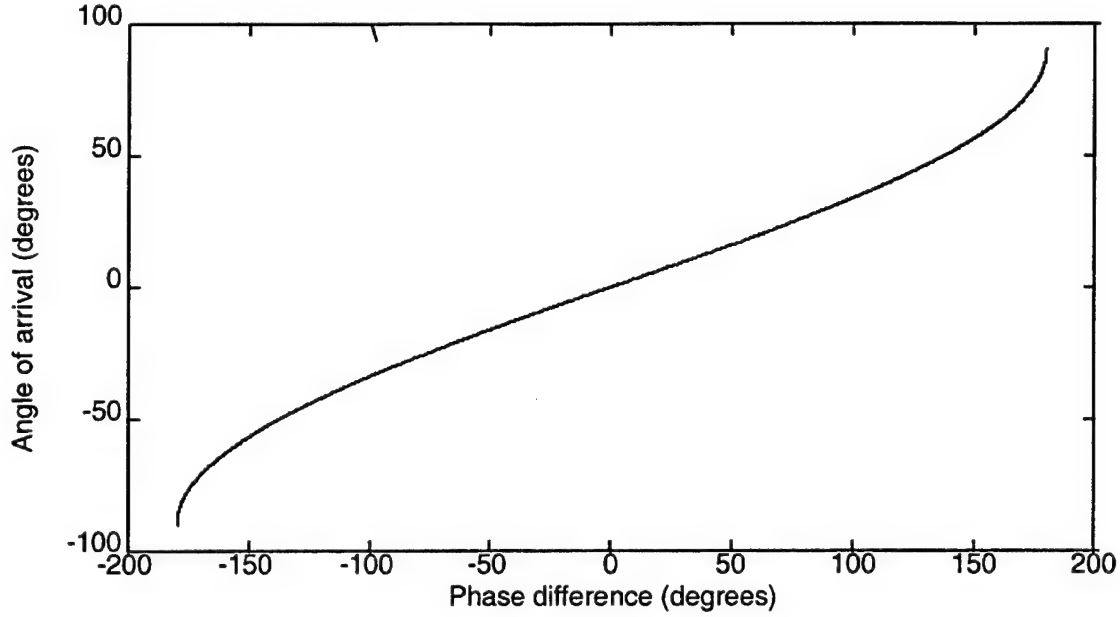


Figure 2.2: Mixer Output Voltage Versus Phase Difference, d=λ/2 [from 6].

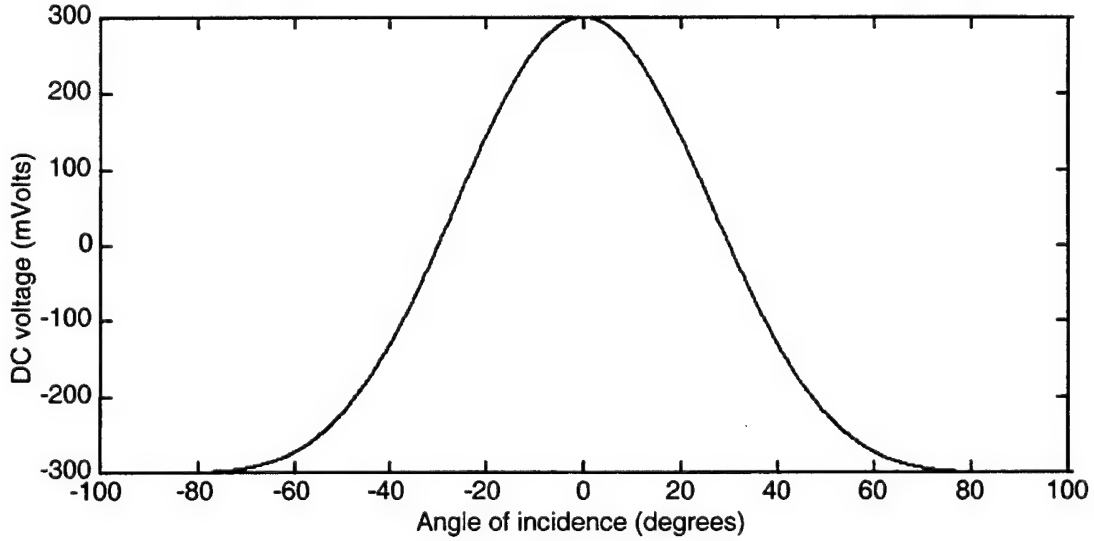


Figure 2.3: Mixer Output Voltage Versus Angle of Incidence, d=λ/2 [from 6].

Ambiguities are generated for baselines where $d > \lambda/2$. That is, the phase detector output voltage is highly ambiguous with a single phase corresponding to many angles of arrival. The number of folding periods n that occur within an AOA of π radians is

$$n = \frac{2d}{\lambda} \tag{2.5}$$

For example, with $d = 7.5\lambda$, n=15 folds are available as shown in Figure 2.4. The folding period is not constant but grows larger in proportion to the angle off of broadside because of the $\sin(\theta_B)$ dependence in (2.4).

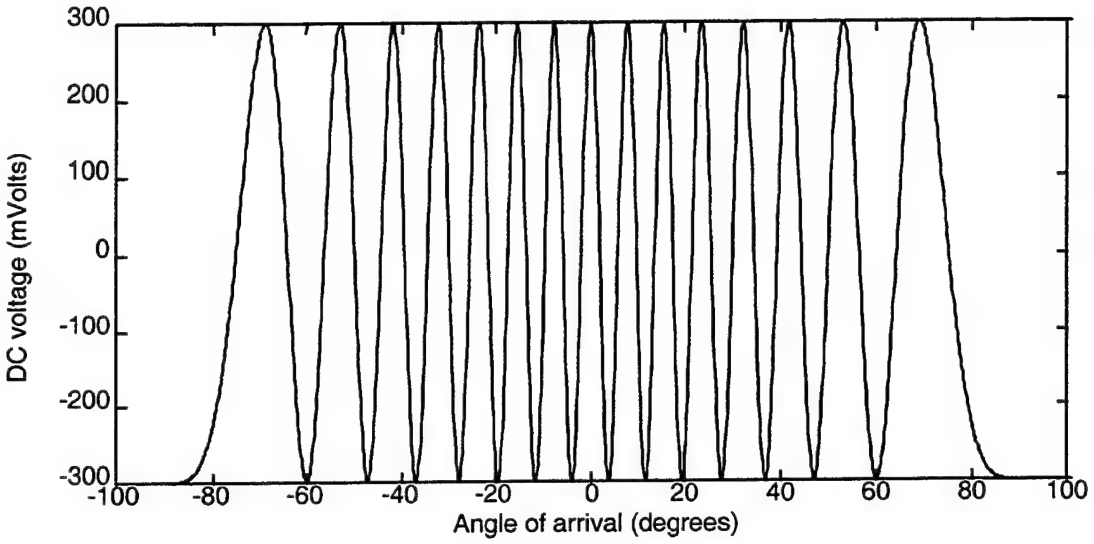


Figure 2.4: Mixer Output Voltage Versus Angle of Arrival for $d = 7.5\lambda$ [from 6].

The ambiguities within the symmetrical folding waveforms represent the phase difference between the elements and can be resolved by using additional interferometers in the linear array. Typically, each interferometer in the linear array symmetrically folds the phase response with the folding period between interferometers being a successive factor of two, or $d_4 = 2d_3 = 4d_2 = 8d_1$, where d_k represents the element spacing. High-speed binary comparators are used to produce a digital output. The folding waveforms are shifted appropriately using a phase shifter in each channel to achieve a Gray code result. The folded output from each phase detector is then quantized with a single comparator with a normalized threshold level T=0.5. Together, the comparator outputs directly encode the signal's AOA in the Gray code format. This approach makes use of the periodic dependence of the interferometer's phase response on the applied plane wave's AOA and the distance between the elements of each interferometer. One of the major

limitations associated with this approach is the achievable resolution. For the folding periods to be a successive factor of 2, the distance between the elements must also be doubled. That is, an 8-bit DF antenna using the previous scheme would require element spacings $\lambda/4,\lambda/2,\lambda,2\lambda,4\lambda\ldots,32\lambda$ with a total baseline length of 32λ. This distance-doubling of the element spacings requires complex analog hardware, adversely affects the physical implementation of the DF architecture and ultimately constrains the achievable resolution.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPTIMUM SYMMETRICAL NUMBER SYSTEM DIRECTION FINDING ANTENNA

A. OPTIMUM SYMMETRICAL NUMBER SYSTEM

Chapter III summarizes Thomas Hatziathanasiou's work in symmetrical number systems [4]. The optimum symmetrical number system (OSNS) is composed of a number of pairwise relatively prime (PRP) moduli m_i. The integers within each OSNS modulus are representative of a symmetrically folded waveform with the period of the waveform equal to twice the modulus, i.e., $2m_i$. For m given, the integer values within twice the modulus are given by the row vector [6, 7]

$$\bar{x}_m = [0,1,\cdots,m-1,m-1,\cdots,1,0] \quad \text{for } m=m_i \ \ i=1, 2, \ldots \tag{3.1}$$

Figure 3.1 shows part of the OSNS folding waveforms and the integer values within the modulus for m_1=5 and m_2=6. The horizontal axis represents the normalized input. The vertical lines represent folding waveform reference levels. The numbers at the top of the figure represent the number of reference levels that are crossed by the folding waveform for a given input value. The period of one complete fold is equal to $2m_1 = 10$ and $2m_2 = 12$.

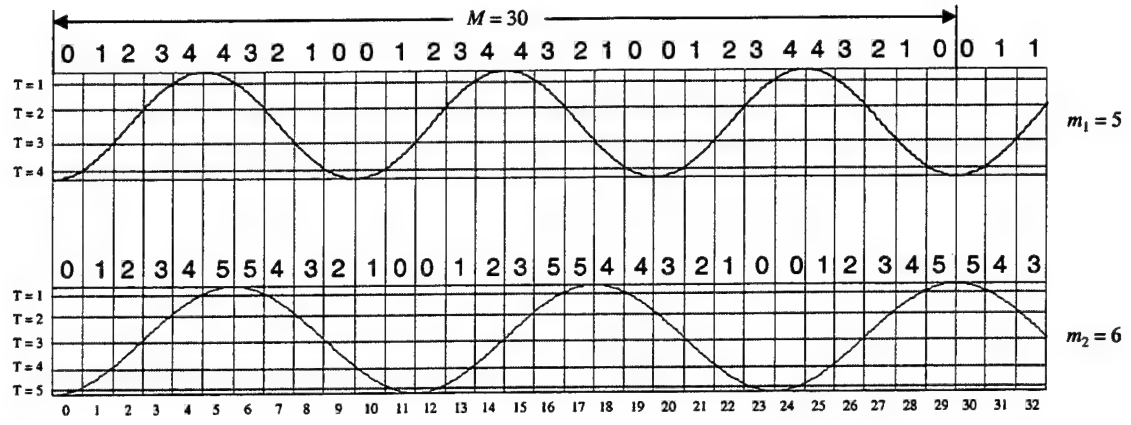


Figure 3.1: Folding Waveforms and Integer Values Within Moduli $m_1 = 5$ and $m_2 = 6$ for the OSNS [from 5].

Due to the presence of ambiguities, the integers within the vector derived by equation (3.1) do not form a complete system of length 2m by themselves. The ambiguities that arise within the modulus are resolved by considering the paired values from all channels together. If N pairwise relatively prime moduli are used, the *dynamic range* (the number of unambiguous vectors) of this scheme is

$$M = \prod_{i=1}^{N} m_i \,. \tag{3.2}$$

This dynamic range is also the position of the first repetitive moduli vector. As shown in Figure 3.1, the dynamic range for the $m_1 = 5$ and $m_2 = 6$ case is M=30. That is, no set appears twice for a normalized input in the range (0,29) [6, 7].

B. OSNS PROTOTYPE DF ANTENNA

Figure 3.2 shows the schematic diagram of a N=2 channel OSNS DF antenna. The array consists of three elements, with all channels sharing a common element. With

the proper distances between elements, each interferometer folds the phase response at $2m_i$. The phase response from each channel is quantized using m_i-1 comparators. The integers shown in Figure 3.1 represent the number of comparators ON due to the phase voltage exceeding the comparator matching threshold voltage.

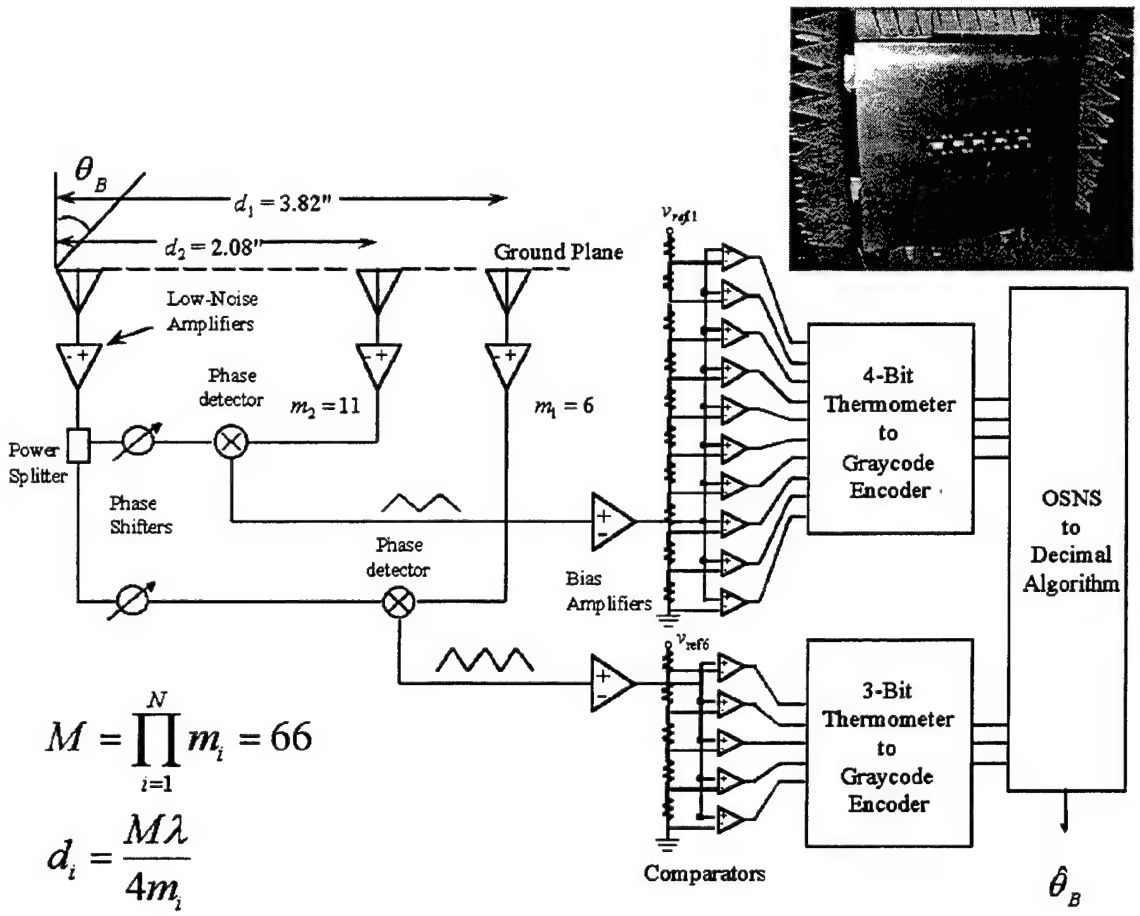


Figure 3.2: Optimum Symmetrical Number System Antenna Architecture [from 4, 8].

The distance between each pair of elements is a function of the modulus, m_i. The phase response must have the correct number of folds to cover the dynamic range. For a given modulus m_i, the number of folds within the dynamic range is

$$n_i = M/2m_i. \tag{3.3}$$

Since each fold of the array corresponds to a change in phase of π for $d = \lambda/2$, the required spacing between the reference element and the respective element of channel i is given by [5]

$$d_i = n_i \frac{\lambda}{2} = \frac{M\lambda}{4m_i}. \tag{3.4}$$

The bin widths are uniform in size in $\sin(\theta_B)$ space with a width 2/M. Therefore, the spatial resolution is given by

$$r_k = \arcsin\left(\xi \frac{2k - M + 2}{M}\right) - \arcsin\left(\xi \frac{2k - M}{M}\right) \tag{3.4}$$

where $k \in \{0,1,\cdots M-1\}$ and k=0 is the bin that starts at $\theta_B = -\pi/2$. The variable ξ is a scale factor that can be used to compress the encoded field of view symmetrically about broadside. The phase waveforms are not ideal over 180 degrees because of the element pattern degradation at wide angles. Since the waveforms at wide angles are not usable, the SNS can be remapped to the usable portion of the waveform to increase resolution. The scale factor is defined to be

$$\xi = \frac{d}{d'} = \frac{\sin(\theta)}{\sin(\theta')}, \tag{3.5}$$

where θ corresponds to phase angle for the unscaled configuration, and θ' is the desired phase angle for the compressed configuration. Note that the spatial resolution is not uniform over the range of $-\pi/2$ to $\pi/2$. The resolution is finer than r at broadside ($\theta = 0^o$) and increases with angle off broadside, because of the $\sin(\theta_B)$ dependence in (2.4).

C. EXPERIMENTAL RESULTS

A linear array based on the moduli $m_1 = 6$ and $m_2 = 11$ was designed, fabricated and tested at a frequency 8.5 GHz [4, 6]. The radiating elements are open-ended waveguides as shown in Figure 3.2. For moduli $m_1 = 6$ and $m_2 = 11$, d_1=3.82 in and d_2=2.08 in. With M=66, the number of folding periods is n_1=5.5 and n_2=3. To provide an adequate signal-to-noise ratio, a low-noise amplifier is included at the output of each interferometer element. Since the common element splits the signal into N paths, an attenuator is placed in the other branches to balance the amplitudes. The amplifiers operate in saturation so that the mixer input signal levels are independent of the angle of incidence. A fixed phase shifter is also included in one branch of each interferometer so that the symmetrically folded phase response waveforms from each mixer may be aligned. This alignment insures that the comparators in the digital processor properly sample the phase waveform and encode it in the OSNS.

The anechoic chamber facility at the Naval Postgraduate School was used for the antenna pattern measurements for each pair of elements. The measured mixer outputs are shown in Figure 3.3 [4, 8]. Since the array is not useful in the endfire regions, the phase response from each channel was minimized at an input AOA of -50°. This is the point where the digital code starts. The locations of the voltage minima are controlled by introducing the appropriate phase shifts to the signals at a point between the elements and the mixer. A simulation of the transfer function for the OSNS array is shown in Figure 3.4 for $\xi = 1$. The measured transfer function (output of digital hardware) is shown in Figure 3.5 [4, 8].

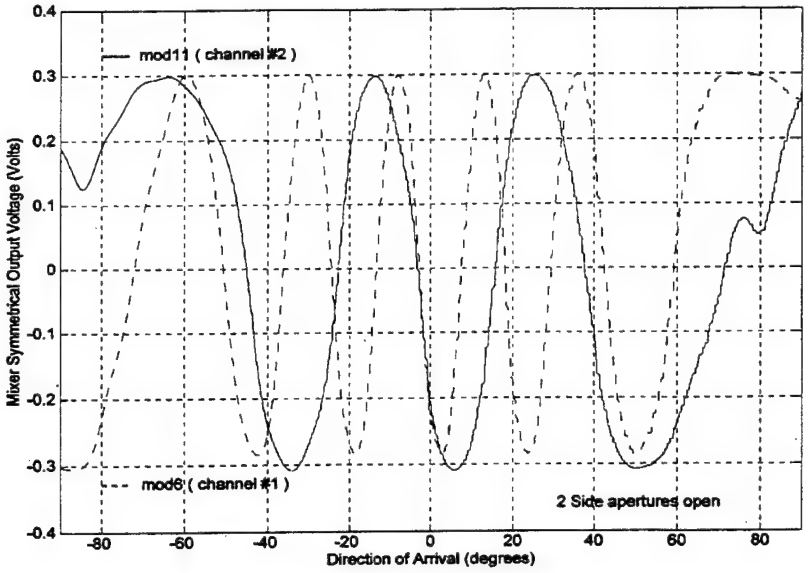


Figure 3. 3: Measured Mixer Output Voltages for OSNS Array [from 4, 8].

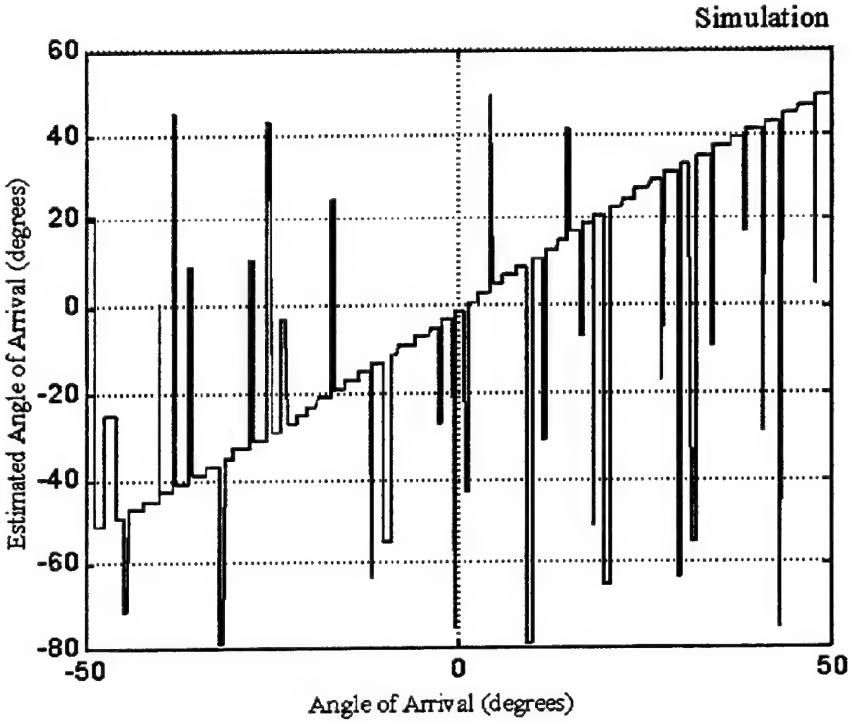


Figure 3. 4: Simulated Transfer Function Using the OSNS Measured Mixer Output [from 4, 8].

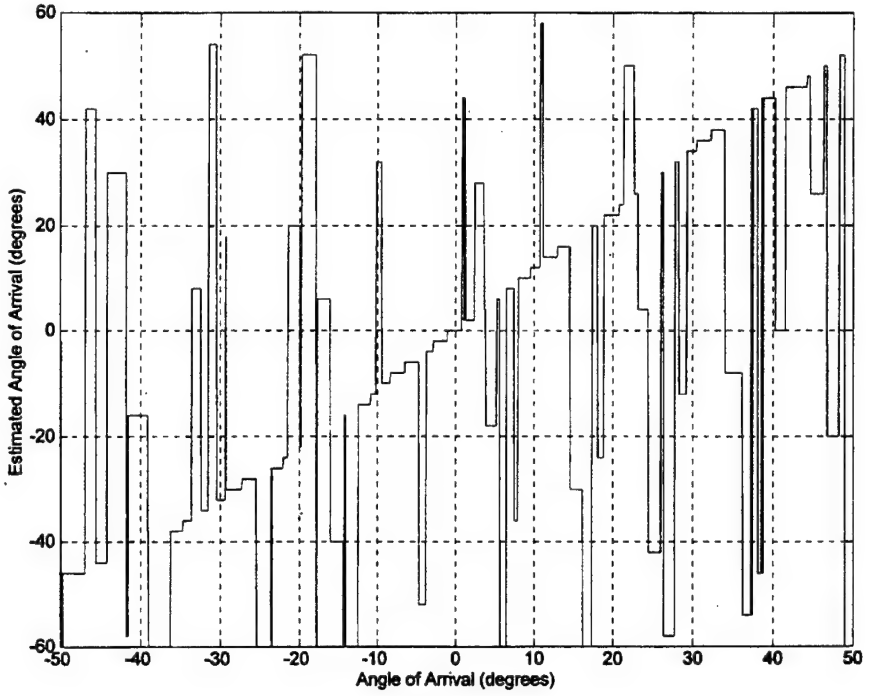


Figure 3. 5: Measured Transfer Function of OSNS Prototype Antenna [from 4, 8].

The spikes in the transfer function are AOA encoding errors. The phase waveforms must cross the comparator thresholds (code transitions) at the same time for all channels. The OSNS can present a problem at each of these specific AOAs. When some comparators, at positions of change, do change, while others do not, the recovered amplitude will have a large error. Encoding errors can also occur due to imperfections in the transmission line phases. These errors however, can be isolated with additional comparators and corrected using interpolation circuits as described in [6]. An alternate approach, however, is to develop a symmetrical number system that eliminates the possibility of this type of encoding error. That is, a symmetrical number system is needed where only one comparator threshold is crossed at any one particular code transition.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ROBUST SYMMETRICAL NUMBER SYSTEM DIRECTION FINDING ANTENNA

A. ROBUST SYMMETRICAL NUMBERING SYSTEM

Chapter IV summarizes David Wickersham's work in symmetrical number systems [5]. In the RSNS, N different periodic symmetrical waveforms are used with pairwise relatively prime integers $m_1, m_2, \ldots, m_N$. The RSNS is based on the following sequence

$$\bar{x}_{ml} = [0, 1, 2, \cdots, m-1, m, m-1, \cdots, 2, 1], \tag{4.1}$$

where $\bar{x}_{ml}$ is a row vector and m is a positive integer ($m > 0$) [9]. In an N-channel RSNS, where $N \geq 2$, the basic sequence for the i^{th} channel (modulus m) is

$$\bar{x}_{ml} = [0, 0, \cdots, 0, 0, 1, 1, \cdots, 1, 1, \cdots, m, m, \cdots, m, m, \cdots, 1, 1, \cdots, 1, 1]. \tag{4.2}$$

In this sequence each value in the $\bar{x}_{ml}$ row vector is put N times in succession. This sequence is repeated in both directions, forming a periodic sequence with the period

$$P_{RSNS} = 2mN. \tag{4.3}$$

Considering a single channel, the discrete states of the robust symmetrical number system can be expressed as [9]

$$g = \begin{cases} \left\lfloor \dfrac{n - s_i}{N} \right\rfloor, & s_i \leq n \leq Nm_i + s_i + 1 \\ \left\lfloor \dfrac{2Nm_i + N - n + s_i - 1}{N} \right\rfloor, & Nm_i + s_i + 2 \leq n \leq 2Nm_i + s_i - 1 \end{cases} \tag{4.4}$$

where g is the n^{th} term of channel i, m_i is the channel modulus, s_i is a corresponding sequence shift $s_i \equiv 0, 1, 2,\ldots, N\text{-}1 \pmod N$ and $N \geq 2$ is the number of channels in the system. The values $\{s_1, s_2,\ldots, s_N\}$ must form a complete residue system modulo N. Because of the relative property of the shifts, one of the shift values will be set equal to 0. The index *n* corresponds to the input value. The discrete states of the RSNS are indexed using the $s_i = 0$ row vector with the index starting from the first zero. Note that the largest integer within each periodic sequence is the modulus m.

An N-channel RSNS is formed of vectors by picking N moduli m_i, and N shift values s_i, $1 \leq i \leq N$. Since the fundamental period for channel i is $2Nm_i$, it follows that the period for the RSNS vectors must be a multiple of $2Nm_i$. Therefore the *fundamental period* for the RSNS is [9]

$$PF_{RSNS} = [2m_1N, 2m_2N,\ldots, 2m_NN], \tag{4.5}$$

where $[a_1, a_2, \ldots, a_N]$ is the least common multiple of $a_1, a_2, \ldots, a_N$. From number theory

$$PF_{RSNS} = 2N[m_1, m_2,\ldots, m_N]. \tag{4.6}$$

To illustrate the RSNS, Figure 4.1 shows the folding waveform and integer values within each modulus for m_1=5 (shift s_1=0) m_2=6 (shift s_2=1). Note the Gray-code properties from one code transition to the next. The thresholds shown on the vertical axis represent the integer values within each RSNS modulus. The integer values occur N=2 times in succession.

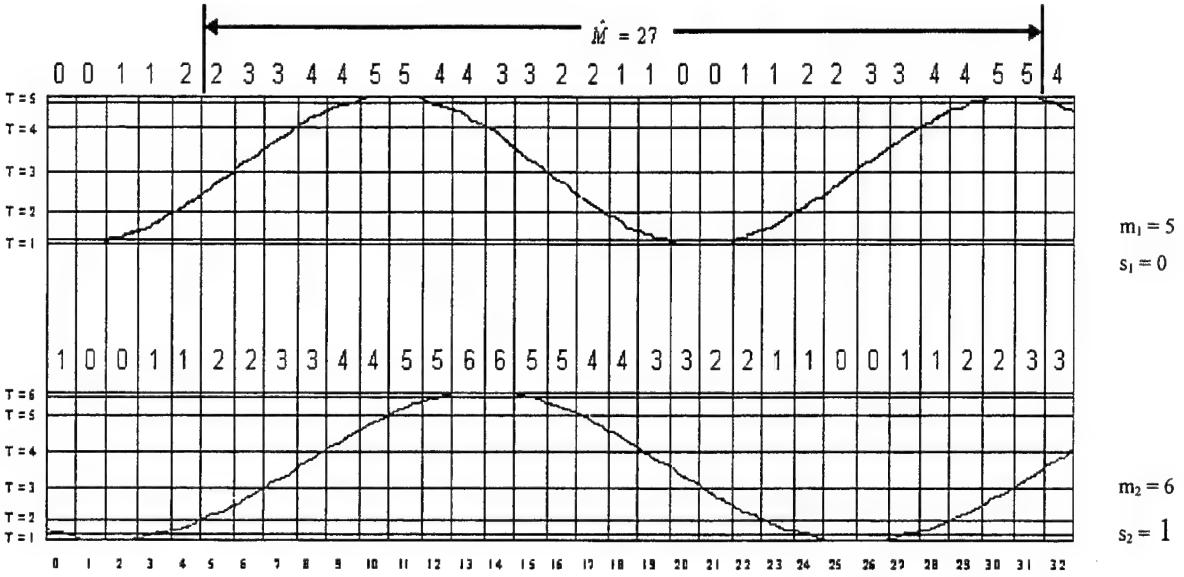


Figure 4.1: Folding Waveforms and Integer Values Within Moduli $m_1 = 5$ and $m_2 = 6$ for the RSNS.

The *system dynamic range* ($\hat{M}_{RSNS}$) of the RSNS is the maximum number of distinct vectors without a redundancy. The selection of the shift and the permutations among the 2 moduli has no effect on $\hat{M}_{RSNS}$. However, the point indices corresponding to $\hat{M}_{RSNS}$ (beginning point and ending point) are different. When the channel moduli are of the form $m_1 = 2^k - 1, m_2 = 2^k, m_3 = 2^k + 1$, the closed-form expression for $\hat{M}_{RSNS}$ is conjectured to be

$$\hat{M}_{RSNS} = \frac{3}{2}m_1^2 + \frac{15}{2}m_1 + 7, \tag{4.7}$$

where $m_1 \geq 3$ [9]. For two channels, with the moduli separated by 3 or more,

$$\hat{M}_{RSNS} = 4m_1 + 2m_2 - 2. \tag{4.8}$$

Checking against computer results when m_1=8 and m_2=17, $\hat{M} = 64$, which fits (4.8). It can be verified that $\hat{M}_{RSNS} = 27 < M = 30$ (3.2) for the two channel example shown in Figure 4.1.

B. RSNS PROTOTYPE ANTENNA

To demonstrate the efficiency of the RSNS preprocessing, the design of a 6-bit antenna using $m_1 = 8$ and $m_2 = 17$ is considered [5]. A schematic diagram of the RSNS antenna is shown in Figure 4.2 for shifts of $s_1=0$, $s_2=1$ resolution bins.

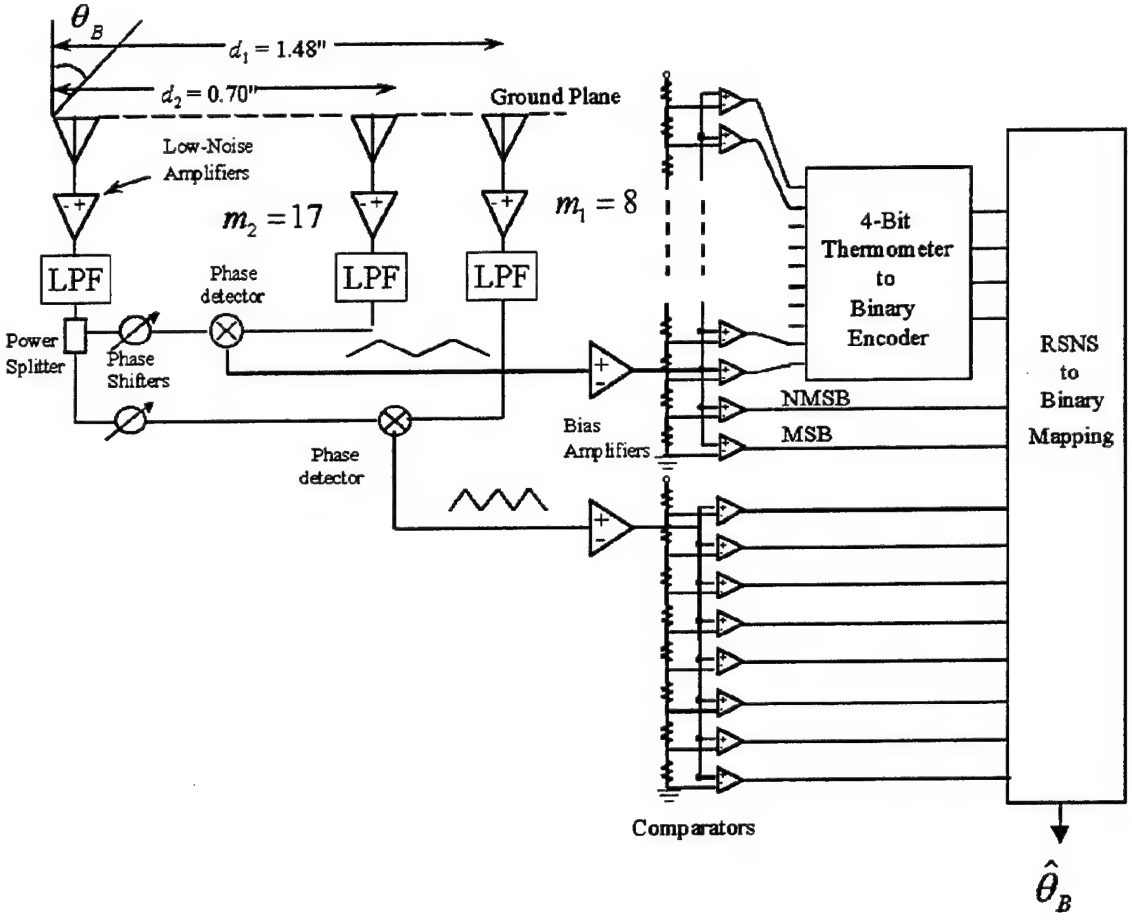


Figure 4.2: Block Diagram of the Unscaled RSNS Antenna [from 5].

The number of folds within each modulus is

$$n_i = \frac{\hat{M}}{P_{RSNS}} = \frac{\hat{M}}{2m_i N}, \tag{4.9}$$

or $n_1=2$ and $n_2=0.94$. The required distance between antenna elements is

$$d_i = n_i \frac{\lambda}{2} = \frac{\hat{M}\lambda}{4m_i N\xi}. \tag{4.10}$$

With a scale factor of $\xi = \sqrt{3}/2$, d_1=1.70 in and d_2=0.80 in. For a normalized folding waveform amplitude (-1 to 1), the threshold for the jth comparator for the modulus m_i channel is [5]

$$V_{j,m_i} = \cos\left(\frac{m_i - j + \frac{1}{2}}{m_i}\pi\right). \tag{4.11}$$

The input signal is applied in parallel to both interferometers. The output from each phase detecting mixer is amplitude analyzed using m_i comparators. For this design, there are a total of 25 comparators with a maximum of 17 loaded in parallel. An RSNS-to-binary logic block translates the RSNS residues (number of comparators ON in each thermometer code) to a more convenient representation. The logic block and corresponding bin number are shown in Figure 4.3.

Vector	Mod 8	Mod 17	Bin	Vector	Mod 8	Mod 17	Bin	Vector	Mod 8	Mod 17	Bin
1	0	0		34	2	16	13	67	1	1	46
2	1	0		35	3	17	14	68	2	1	47
3	1	1		36	2	17	15	69	2	0	48
4	2	1		37	2	16	16	70	3	0	49
5	2	2		38	3	16	17	71	3	1	50
6	3	2		39	3	15	18	72	4	1	51
7	3	3		40	4	15	19	73	4	2	52
8	4	3		41	4	14	20	74	5	2	53
9	4	4		42	5	14	21	75	5	3	54
10	5	4		43	5	13	22	76	6	3	55
11	5	5		44	6	13	23	77	6	4	56
12	6	5		45	6	12	24	78	7	4	57
13	6	6		46	7	12	25	79	7	5	58
14	7	6		47	7	11	26	80	8	5	59
15	7	7		48	8	11	27	81	8	6	60
16	8	7		49	8	10	28	82	7	6	61
17	8	8		50	7	10	29	83	7	7	62
18	7	8		51	7	9	30	84	6	7	63
19	7	9		52	6	9	31	85	6	8	
20	6	9		53	6	8	32	86	5	8	
21	6	10	0	54	5	8	33	87	5	9	
22	5	10	1	55	5	7	34	88	4	9	
23	5	11	2	56	4	7	35	89	4	10	
24	4	11	3	57	4	6	36	90	3	10	
25	4	12	4	58	3	6	37	91	3	11	
26	3	12	5	59	3	5	38	92	2	11	
27	3	13	6	60	2	5	39	93	2	12	
28	2	13	7	61	2	4	40	94	1	12	
29	2	14	8	62	1	4	41	95	1	13	
30	1	14	9	63	1	3	42	96	0	13	
31	1	15	10	64	0	3	43	96	0	14	
32	0	15	11	65	0	2	44	97	1	14	
33	0	16	12	66	1	2	45	98	1	15	

Figure 4.3: RSNS Logic Block [from 5].

A consequence of quantizing the angle of arrival into a bin is that the RSNS system reports any signal that falls within a bin as if it arrived at the bin center, $\hat{\theta}_k$. That is to say, for one angle within each bin, the estimation is exact, but for the remaining angles a reporting error exists. The AOA resolution or bin width in *θ-space* for the kth bin is

$$r_k = \arcsin\left(\xi\frac{2k-\hat{M}+2}{\hat{M}}\right) - \arcsin\left(\xi\frac{2k-\hat{M}}{\hat{M}}\right). \tag{4.12}$$

Another advantage of the RSNS is that small phase errors can be tolerated without a serious error to a reported AOA. These phase errors cause the folded waveform from the mixer to shift, thus changing the timing of the comparator state changes. Because the waveforms from the mixers are quantized, the bin width or r_k provides some free-play in the antenna circuitry and limits AOA reporting errors to the adjacent quantization levels (bins). However, there is a limit to this feature. If the phase error becomes too great, the comparators will not change state in the correct sequence, which may or may not correspond to a code that is mapped in the dynamic range. These errors are similar to the errors that the OSNS experiences when the comparators of both channels do not change state at the same time. For the system designed, a theoretical phase error of 6° can be tolerated in the modulus 17 channel and a phase error of 12° in the modulus 8 channel. Figures 4.4, 4.5 and 4.6 show the simulated transfer function. Figure 4.4 is the transfer function with no phase error in both channels. Figures 4.5 and 4.6 are the modulus 8 channel with zero degrees of phase error and 3° and 6° phase error in the modulus 17 channel. Note how the transfer function becomes jagged and erratic as the phase error is increased until it becomes discontinuous at 6°.

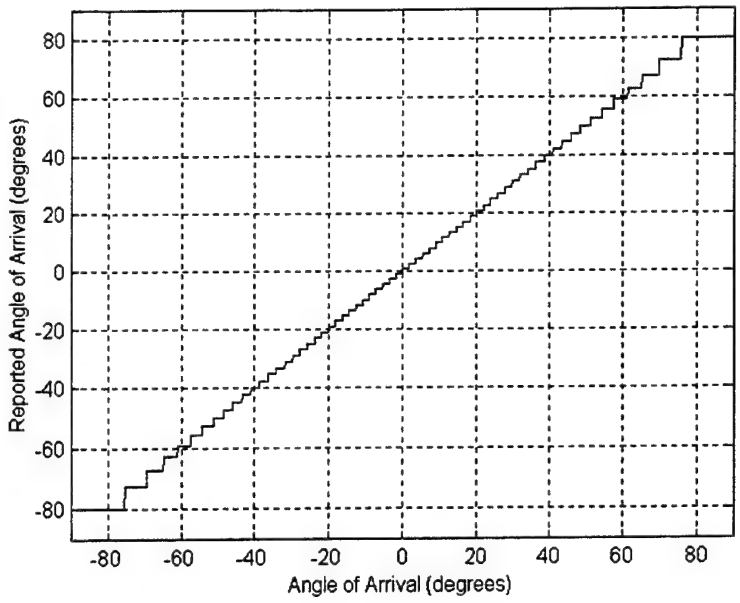


Figure 4.4: Simulated Transfer Function with No Phase Errors in Either Channel [from 5].

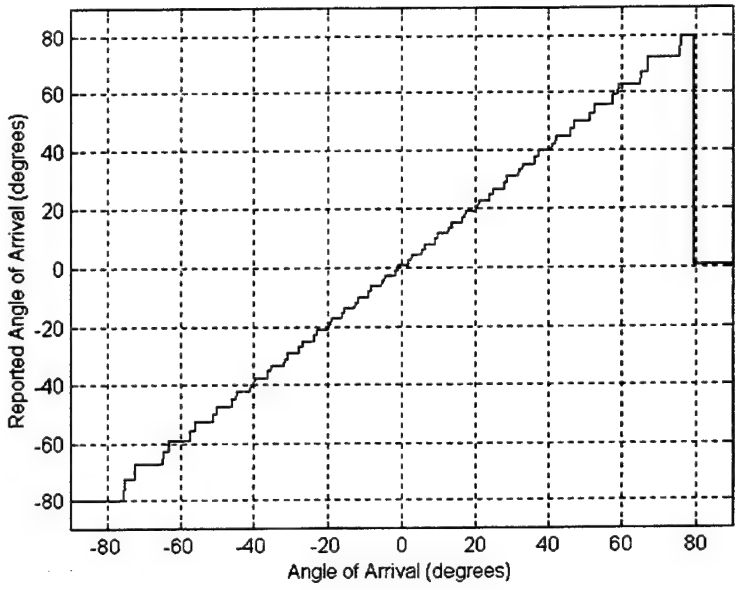


Figure 4.5: Modulus 17 Channel with 3° Phase Error Introduced [from 5].

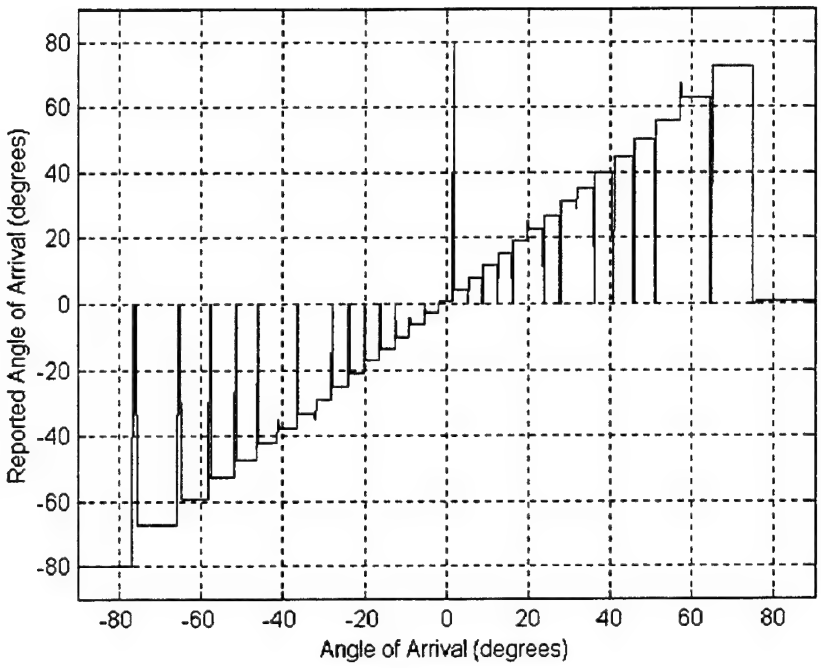


Figure 4.6: Modulus 17 Channel with 6° Phase Error Introduced [from 5].

C. EXPERIMENTAL RESULTS

A RSNS array based on the moduli $m_1 = 8$ and $m_2 = 17$ was designed, fabricated and tested at a frequency of 8.0 GHz [5]. The schematic diagram of the two-channel array is shown in Figure 4.2. The radiating elements are printed circuit dipoles. For moduli $m_1 = 8$ and $m_2 = 17$, the unscaled distances, $\xi = 1$, are d_1=1.48 in. and d_2=0.70 in. With $\hat{M} = 64$, the number of folding periods n_1=2 and n_2=0.94. To provide an adequate signal-to-noise ratio, a low-noise amplifier is included at the output of each interferometer element. Also included in the RSNS array is a lowpass filter in each channel to eliminate the amplifier harmonics that are present. Since the common element splits the signal into N paths, an attenuator is placed in the other branches to balance the amplitudes. A fixed phase shifter is also included in one branch of each interferometer so that the symmetrically folded phase response waveforms from each mixer may be aligned. This alignment insures that the comparators in the digital processor properly

sample the phase waveform and encode it in the RSNS. For the $m_1 = 8$ channel, 8 comparators are required. For the $m_2 = 17$ channel 17 comparators are required. The EEPROM that was used accommodates 15 inputs. The modulus 17 channel processing is conducted in two stages. In the first stage, the 4-bit thermometer to binary encoder maps the 15 least significant codes out of the 17 comparators (ones with the smallest threshold). The remaining two comparators are mapped on the second EEPROM along with the 4 bit binary encoder output and all modulus 8 comparators. The digital processor puts out a bin number k corresponding to the estimated AOA $\hat{\theta}$. The estimated AOA for bin k is given as

$$\hat{\theta}_k = \sin^{-1}\left(\xi \frac{2k+1}{\hat{M}} - \xi\right). \tag{4.13}$$

Broadside incidence for the RSNS antenna corresponds to the transition between Bins 31 and 32. For the modulus 17 channel at slightly negative angles (bin 31), the ninth comparator is ON and for slightly positive angles (bin 32), the ninth comparator is OFF. That is, the phase value at $\theta_B = 0$ corresponds exactly to the threshold of the ninth comparator. From (4.11) the phase corresponding to the threshold value $V_{17,9}$ is $\pi/2$. This is the phase value required for the modulus 17 channel at $\theta_B = 0$ and is obtained using the phase adjuster shown in Figure 8. In the modulus 8 channel, $\theta_B = 0$ occurs midway between comparator 7 and 6 thresholds. Since $V_{8,7} = \cos^{-1}\left(\frac{5\pi}{16}\right)$ and $V_{8,6} = \cos^{-1}\left(\frac{3\pi}{16}\right)$, the required phase value is $\frac{4\pi}{16} = \frac{\pi}{4}$ and is also obtained using the modulus 8 phase adjuster.

The normalized mixer folding waveform outputs are shown with the simulated waveforms in Figures 4.7 for the m_1=8 channel and 4.8 for m_2=17 channel for $\xi = 1.0$. The basic features correspond to predicted curves. The measured folding outputs from the shift and bias amplifiers in both channels are shown in Figure 4.9. The digital circuit maps the phase difference at the mixer output into an estimated angle of arrival. The predicted output of the unscaled prototype array is reproduced as Figure 4.10. The

measured mixer output contains phase errors, which result in differences between the simulated and predicted data. These phase errors cause the thermometer code to incorrectly map the incident wave to the proper angle of arrival. The measured transfer function of the unscaled prototype antenna is shown in Figure 4.11.

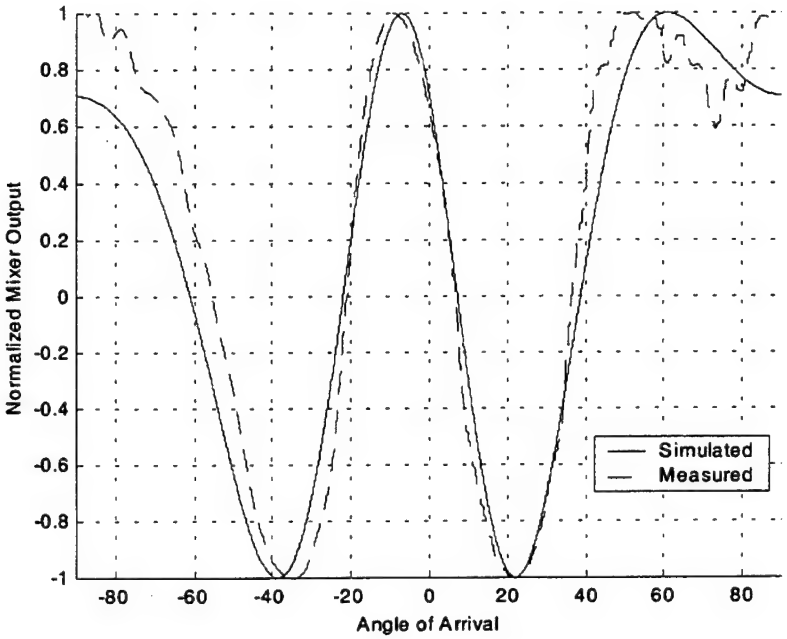


Figure 4.7: Normalized Mixer Output and Simulation Waveforms for RSNS Array with m_i=8 [from 5].

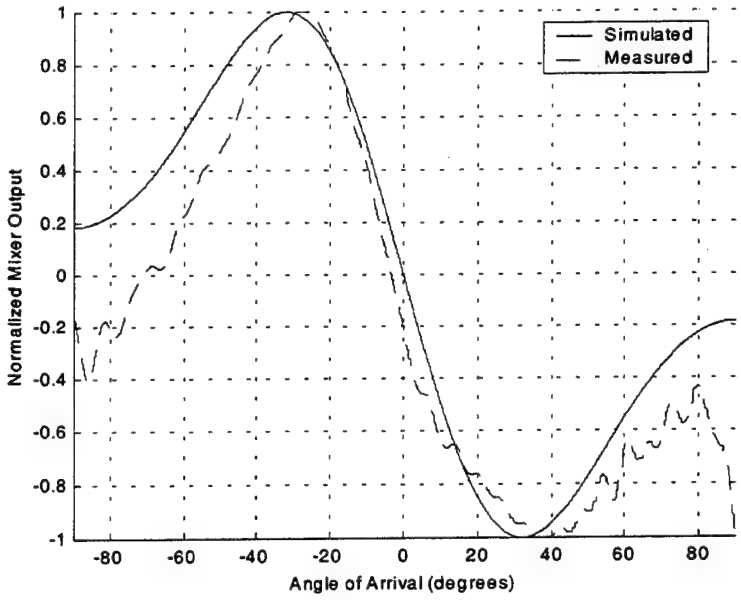


Figure 4.8: Normalized Mixer Output and Simulation Waveforms for RSNS Array with m_2=17 [from 5].

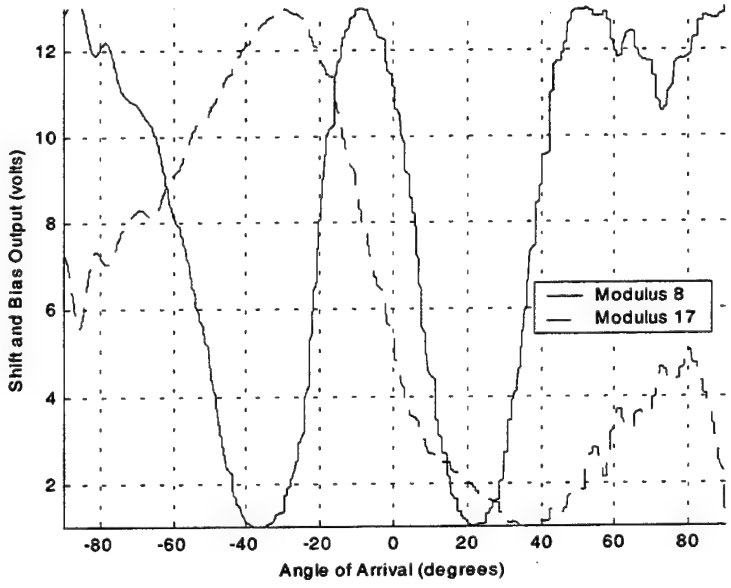


Figure 4.9: Measured Folding Waveform Outputs from the Shift and Bias Amplifiers [from 5].

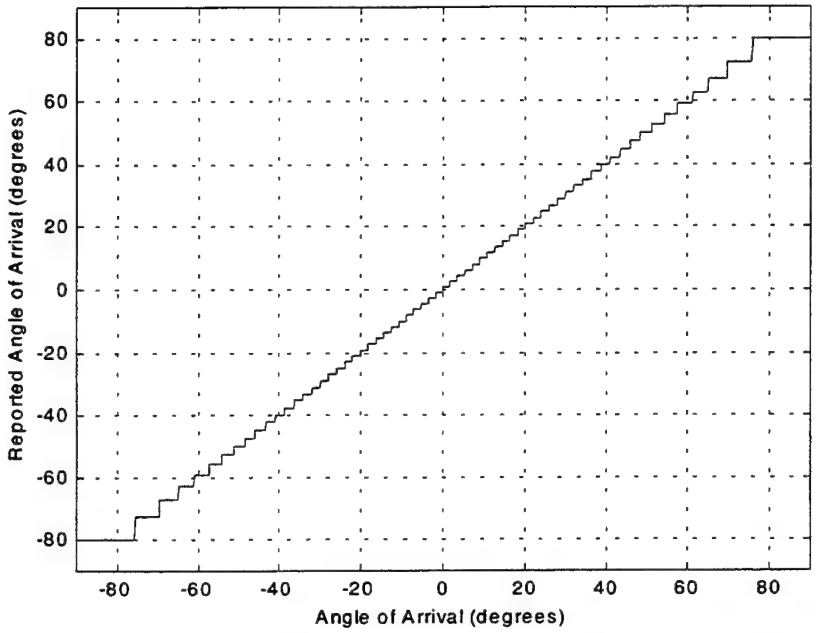


Figure 4.10: Predicted RSNS Transfer Function [from 5].

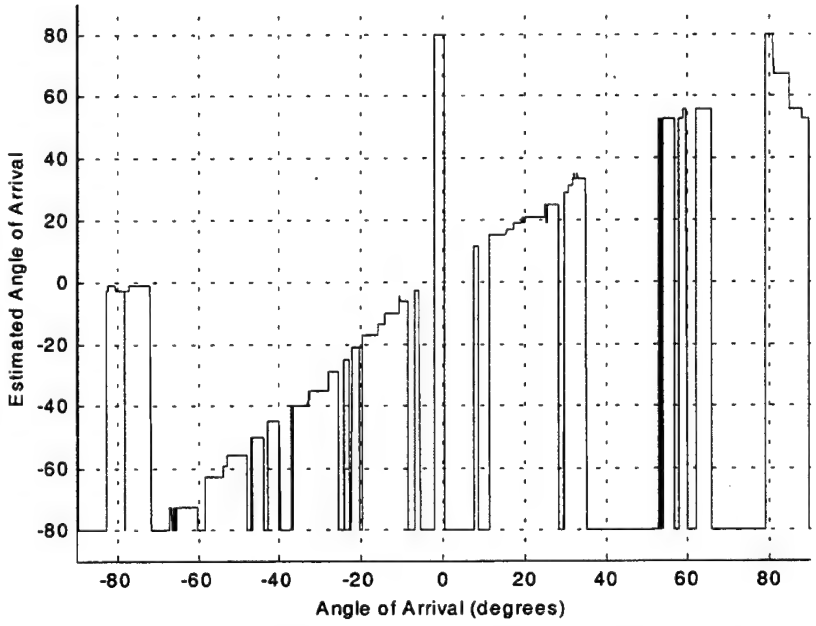


Figure 4.11: Measured RSNS Transfer Function [from 5].

The large discontinuities in the transfer function are due to phase and amplitude errors, which arise from several sources. Mutual coupling is a common cause of phase

front distortion in small arrays of closely spaced elements. In order to increase the distance between elements and thus reduce the mutual coupling, a scale factor ξ can be introduced. The scale factor is the ratio of the element spacings of the scaled and unscaled arrays. Reducing the scale factor narrows the mapable field of view of the antenna and increases the resolution within the mapable field of view. For a scale factor of $\xi = \sqrt{3}/2$, the maximum mapable aperture (θ_{MM}) is 60°. The predicted transfer function for the scaled array is shown in Figure 4.12. The corresponding measured transfer function is reproduced as Figure 4.13. The mixer outputs contain significant phase errors, and the increased resolution of the antenna limits the ability to process the errors. The steeper slope of the scaled transfer function is clearly visible when comparing 4.10 and 4.12.

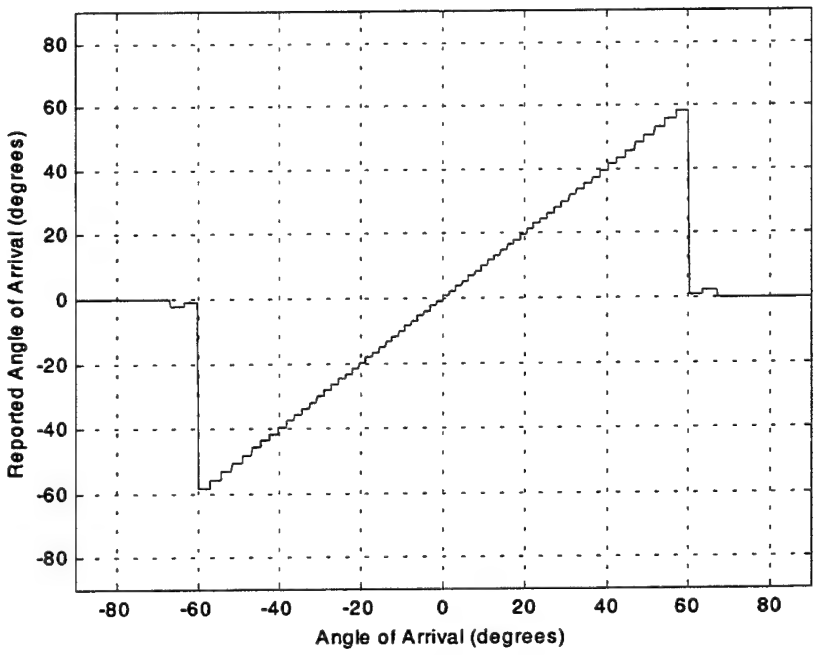


Figure 4.12: Predicted RSNS Transfer Function for Scale Factor $\xi = \sqrt{3}/2$ [from 5].

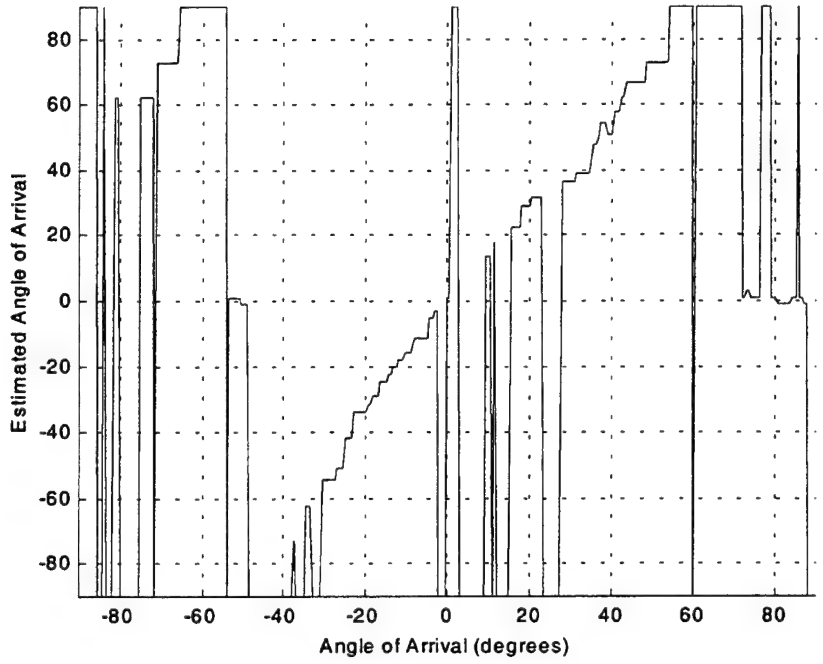


Figure 4.13: Measured RSNS Transfer Function for Scale Factor $\xi = \sqrt{3}/2$ [from 5].

The transfer function for the scaled array of dipoles does not show any significant improvement due to the reduction of mutual coupling. To verify this, a second array comprised of open-ended waveguide elements was constructed and tested. These waveguide elements have significantly less H-plane mutual coupling than the dipoles, but the transfer function showed no improvement over with printed circuit (dipole) elements. This indicates that mutual coupling is not the primary cause of the phase errors. The microwave circuit has three likely sources of errors: (1) the low noise amplifiers, (2) coaxial cables and phase adjusters, and (3) the rigid connectors. The prototype uses two cascaded LNAs operating in saturation in each channel. Harmonics generated by the first amplifier are amplified by the second amplifier. The second amplifier also introduces its own harmonics. Filters were used to suppress harmonics.

The coaxial cables and phase adjusters are another source of error within microwave circuits. The rigid coaxial cables are coarsely trimmed at 8.0 GHz, and the phase shifters are used for fine trimming. The cable lengths are long and subject to thermal expansion and contraction, particularly in the vicinity of the amplifiers. Finally, all of the devices in

the circuit are joined with SMA type connectors. The circuit was repeatedly assembled and disassembled for troubleshooting, and consequently the quality of the connections were degraded. Phase errors of $\pm 2°$ can easily be introduced at each connection. Thus it appears that many small errors accumulate in a manner such that the 6° and 12° error thresholds are exceeded, which results in a degraded transfer function.

V. IMPROVED RSNS RF SYSTEM DESIGN

Previous research has introduced the Optimal Symmetrical Number System and the Robust Symmetrical Number System and shown that they are able to efficiently resolve ambiguities in phase sampling interferometry. The prototype arrays that were built, however, both showed that significant errors are introduced due to the nature of the components and construction of the system. Non-linear components, poor connections and mutual coupling are prime contributors to the phase errors present in the folding waveforms. This thesis research takes a step forward to validate the RSNS design and to eliminate the phase errors in the system.

The RF circuit shown in Figure 5.1 receives a transmitted signal with a set of micro-strip antennas (see Figure 4.2). The signal is passed through a set of isolators and then to the first amplification stage. Band-pass filters are used to remove harmonics and the signal is then sent to the second amplification stage. Low-pass filters are used to remove harmonics from the second stage amplifiers. A power splitter is used to divide the reference signal that is passed on to the RF side of the balanced mixers. The signals from each channel are passed through phase shifters and then to the local oscillator (LO) side of the mixers. Since the frequency at the RF and LO inputs are the same, the mixer output is a DC value that is a function of the AOA. This DC function is referred to as the folding waveform that is amplitude analyzed by the RSNS signal processor in order to determine the AOA.

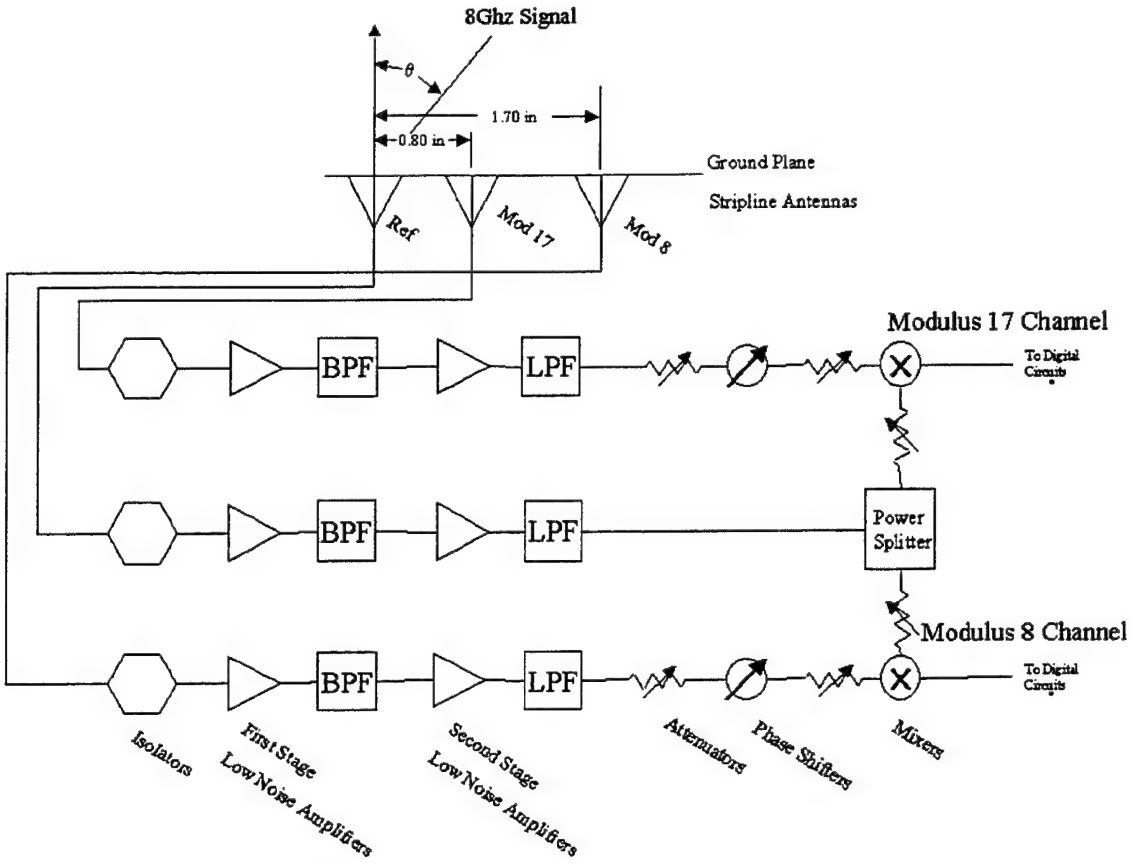


Figure 5.1: RF Circuit Block Diagram.

A. ANTENNA ELEMENTS

The open-ended waveguide antennas that were used in the OSNS design are too wide to be used in the RSNS array. Since the element spacing for the RSNS array is small, micro-strip antennas were developed to replace the open-ended waveguide antennas. The stripline antennas are half-wave dipoles designed to resonate at 8.0 GHz. A schematic diagram of a micro-strip dipole antenna is shown in Figure 5.2. The width of each element is only 60 mils. The elements are placed into a linear array with the spacings set at 0.80" between the reference and modulus 17 elements and 1.70" between the reference and the modulus 8 elements as shown in Figure 5.3. These spacings were chosen over the original design spacings of 0.70" and 1.48", since the folding waveform

becomes distorted at angles greater than 60° from broadside. By compressing the RSNS dynamic range into the reduced field of view a higher angular resolution is obtained. An additional benefit of the increased spacing is that any possible mutual coupling effect is decreased. Antenna pattern measurements were taken of this array one element at a time; both phase and power were recorded. Figures 5.4 and 5.5 show the results respectively.

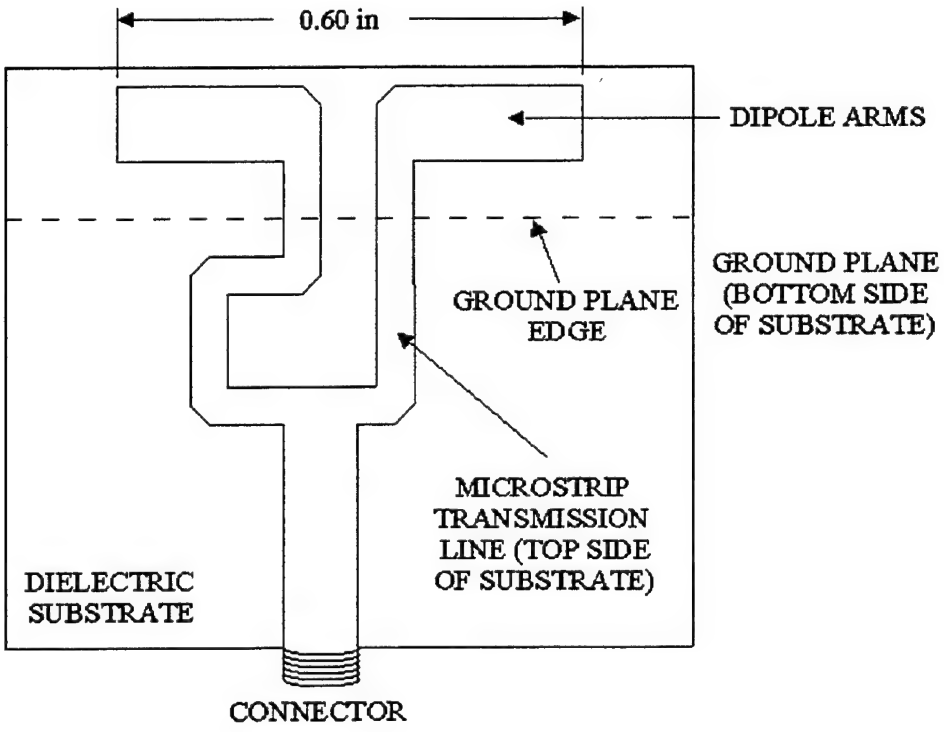


Figure 5.2: Micro-strip Dipole Antenna Element [from 5].

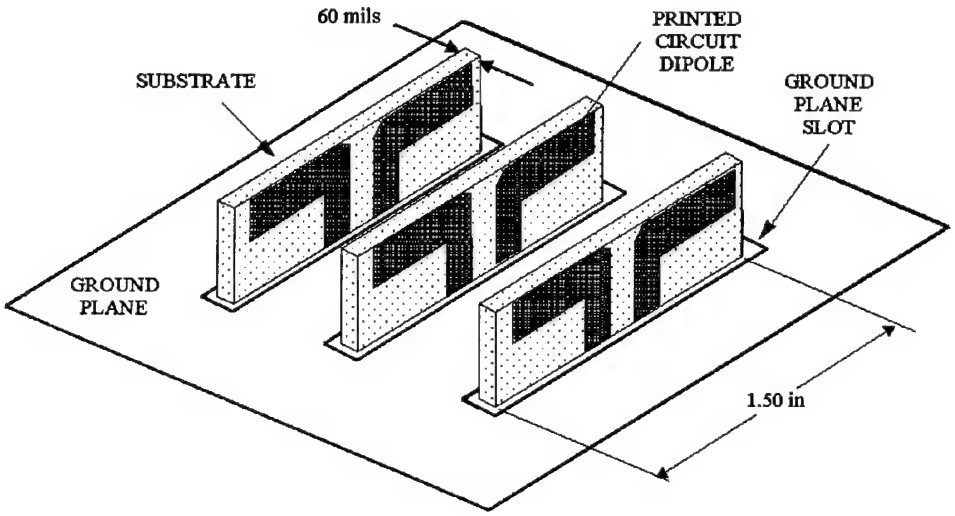


Figure 5.3: Stripline Antenna Array [from 5].

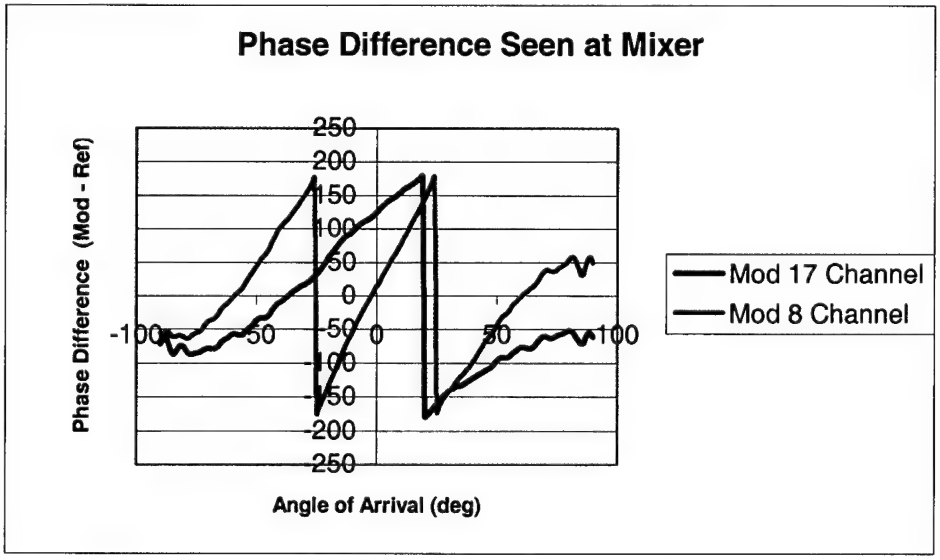


Figure 5.4: Phase Difference Between Modulus and Reference Signals.

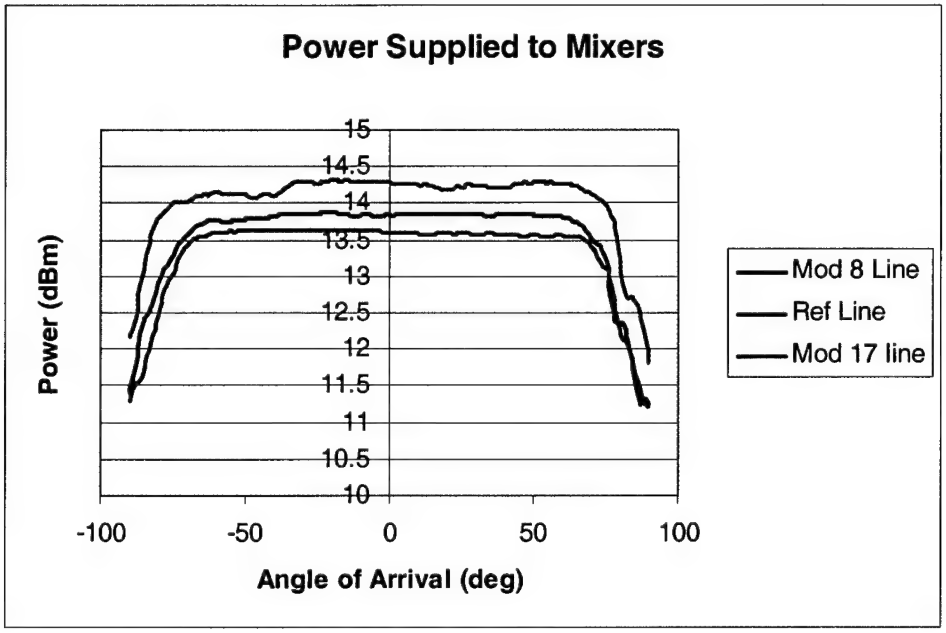


Figure 5.5: Power Measured at the Mixer Inputs.

B. ISOLATORS

M2 Global S70124 isolators were used to further reduce mutual coupling effects. They are placed between the antenna elements and the first amplification stage. The specifications for the isolators are shown in Table 5.1.

Frequency Range (GHz)	Isolation (dB) Minimum	Insertion Loss (dB) Maximum	VSWR Maximum	Temperature Range (°C)	Connector Available	Package Size (in)
7.0 - 12.4	20	0.5	1.25	-20 to +60	SMA	.75 x .90 x .50

Table 5.1: Isolator Specifications.

After testing the isolators on the network analyzer they were found to meet all the specifications that were supplied.

C. AMPLIFIERS

The system was first designed with only one stage of amplifiers. The one amplifier design was introduced to try and eliminate sources of phase errors that were present in the first prototype. The DBS DWT 18636 amplifiers were used. The goal of the amplification stage is to gain a constant power output to send to the mixers since any mismatch in power at the mixer inputs will produce phase errors. These microwave amplifiers have two operating regions: linear and saturation. The preferred region for this system is the saturated region. By operating in saturation, the output power of the amplifiers is independent of the input power. Figure 5.6 shows the power curves for the DBS amplifiers.

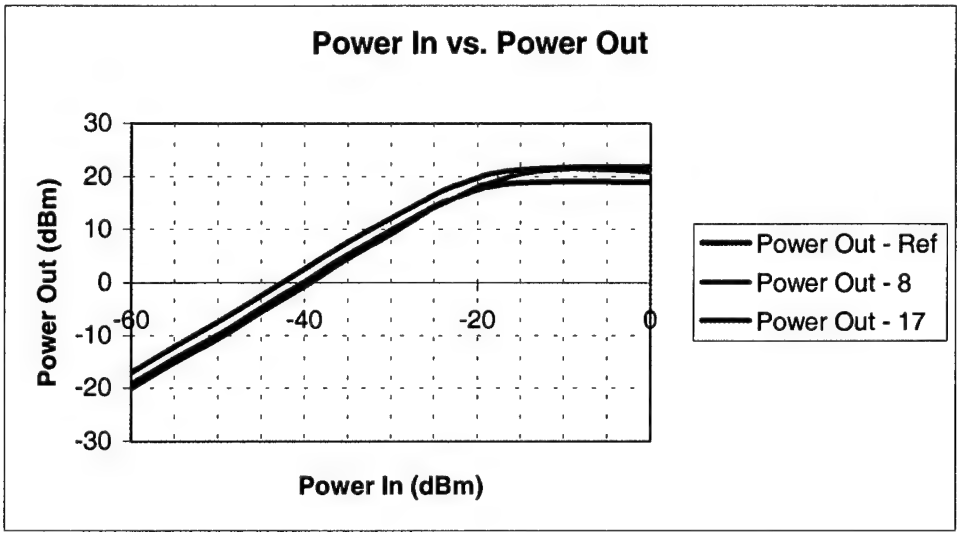


Figure 5.6: Power Curves for DBS Amplifiers.

In order for the system to work well the input power needs to be at least -15 dBm. The initial transmitter distance was 19 feet. The input power for the amplifiers at this distance was not enough to drive the amplifiers into saturation. A stand was constructed to move the transmitter closer to the antenna elements. However, even with the transmitter at 6' from the elements, the power input to the amplifiers was only -17 dBm. This level was barely enough to saturate the amplifiers, so a two-stage amplifier design was utilized.

The DBS amplifiers are used in the second stage of amplification because the power output levels are well matched and the three were phase matched to within 10°.

The first stage amplifiers were chosen based on their power curves. The two primary considerations were maximum gain and how well the power curves matched. Figure 5.7 shows the power curves for the three Avantek amplifiers that were chosen.

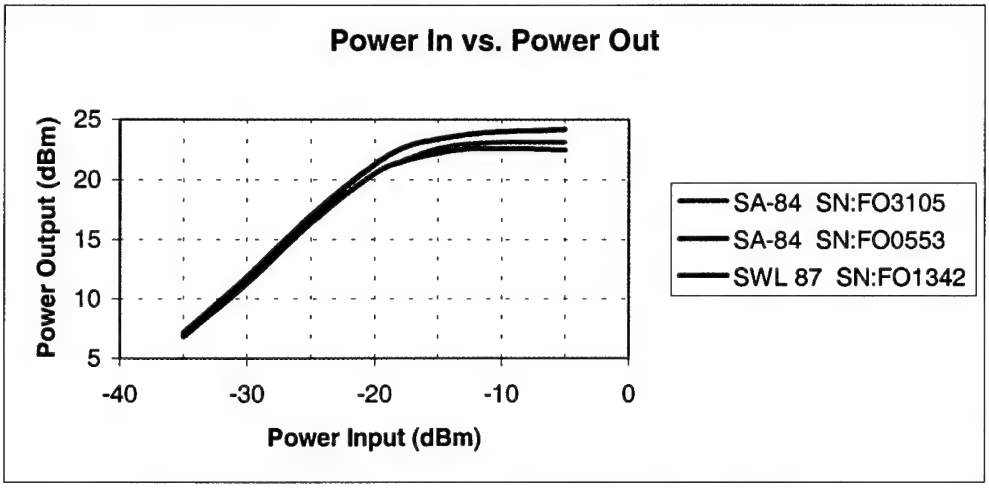


Figure 5.7: Power Curves for Avantek Amplifiers.

Phase matching for this amplification stage is not as important as the second stage. The purpose for the first stage is to drive the second stage amplifiers into saturation. The gain for these amplifiers is approximately 40 dB in the linear region, which is more than enough to saturate the second stage amplifiers at a received power of -50 dBm. The overall effect of the two stages is that the signal output from the second amplification stage remains a constant regardless of the received power by the antenna elements.

D. FILTERS

The amplifiers used in this system produce harmonics. The harmonic levels are 20 dB lower than the fundamental, but introduce phase errors in the system in excess of 20°. In order to eliminate these errors, filters are used after each stage of amplification. AIRTRON band-pass filters, with a pass band of 4-8 GHz, are used after the first stage amplifiers and MICROLAB/FXR LA-90N low-pass filters are used after the second stage

amplifiers. Each filter has an attenuation of 55 dB or greater in the stopband, which effectively eliminates all signals except the fundamental.

E. MIXERS

A mixer is a common microwave component that is used to convert a higher RF into an intermediate frequency (IF). Typically, the RF signal is combined with a second signal, referred to as the local oscillator (LO), at a non-linear diode. The resulting IF is approximated by $\omega_{if} = \omega_{rf} - \omega_{lo}$ [10]. The $\omega_{rf} + \omega_{lo}$ and other product terms that are generated in the mixer are removed by filtering. For the RSNS antenna, the mixers are used as phase detectors. The RF and LO inputs are the same frequency, therefore, the IF output is a DC value.

$$IF = A_1 \cos[(\omega_{lo} - \omega_{rf})t - (\phi_{lo} - \phi_{rf})] + A_2 \cos[(\omega_{lo} + \omega_{rf})t - (\phi_{lo} - \phi_{rf})] + [\text{higher frequency terms}]. \quad (5.1)$$

Since $\omega_{rf} = \omega_{lo}$,

$$IF = A_1 \cos(\phi_{rf} - \phi_{lo}) + A_2 \cos(2\omega t) + (\text{higher frequency terms}). \quad (5.2)$$

With proper filtering the $2\omega t$ and higher frequency terms can be eliminated. The remaining DC term depends solely on the phase difference $(\phi_{rf} - \phi_{lo})$,

$$IF = A_1 \cos(\phi_{rf} - \phi_{lo}) . \quad (5.3)$$

Two mixers were considered for this system, the Mini-Circuits ZMX-10G mixer and the Anaren 70540 balanced mixer. Table 5.3 lists the specifications for each mixer.

Model	Frequency	IF Frequency	Conversion Loss	LO-RF Isolation
	GHz	MHz	dB Max	dB Min
Mini-Circuits	3.7-10.0	DC-2000	8.5	20.0
Anaren	5.2-10.4	DC-1100	7.5	6.0

Table 5.2: Mixer Specifications.

For this application, the important specification is the IF frequency range. By having the range set below 2ω it effectively filters the output of the mixer.

The next important parameter that needs to be examined is the phase response of the mixer. Figure 5.8 shows the phase response of both mixers when connected to the network analyzer.

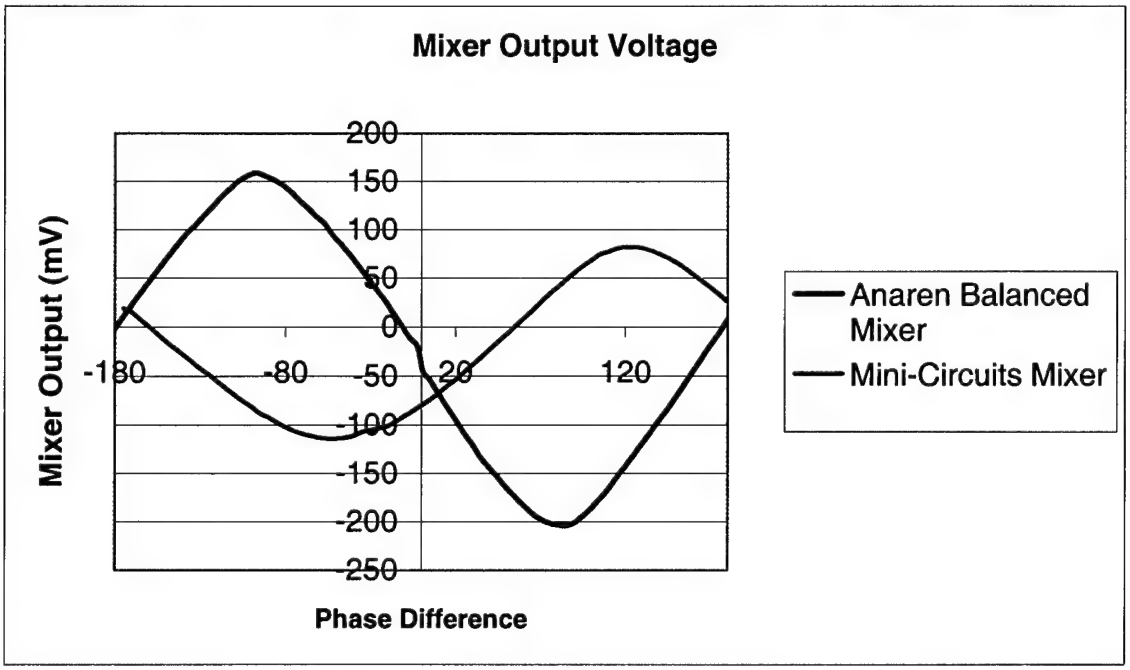


Figure 5.8: Mixer Phase Response.

By comparing the curves, the Anaren balanced mixer was chosen since the Anaren mixer's response has a more symmetrical shape about the 0° axis and has its maximum and minimum values at ± 90°. Also noticeable is a DC offset of approximately -40 mV. This offset can be compensated in the level and bias circuits at the input of the signal processing circuit. One drawback that was discovered during testing is the Anaren mixer

has a low saturation level. Figure 5.9 shows the first set of folding waveforms with the power level to the inputs at 14 dBm.

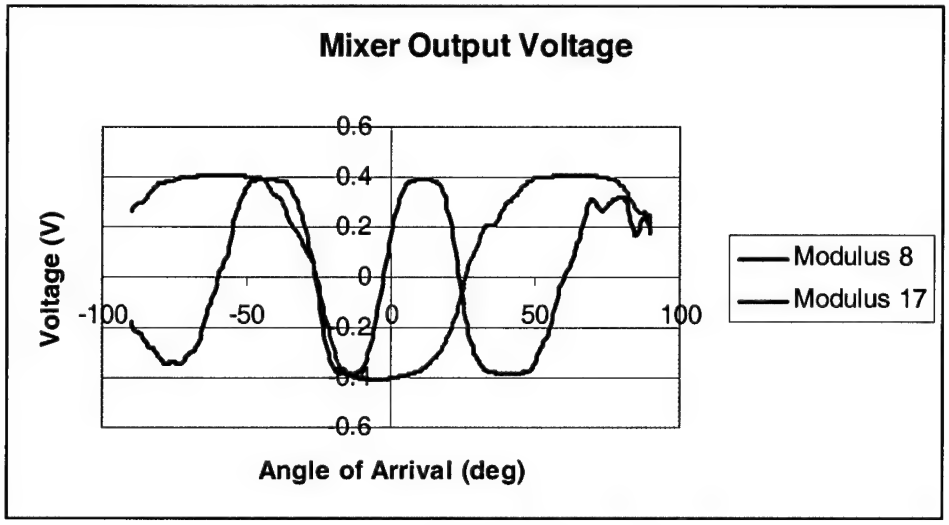


Figure 5.9: Folding Waveforms from Saturated Mixers.

The maximum and minimum regions of the folding waveforms appear to be broadened, indicating that the mixer is saturated. Attenuators were added at the mixer inputs until the regions regained their predicted shape.

F. MISCELLANEOUS COMPONENTS

The phase difference between the reference signal and the channel signal is the basis for AOA estimations. The phase difference at broadside defines the relationship of the folding waveforms. In order to set the phase angles in each of the channels, the lengths of the semi-rigid coaxial cable were precisely cut. Because predicting how the phase of the signal is affected by the amplifiers and other components is difficult, phase adjusters were added. The phase adjusters are variable length coaxial cables that contain no dielectric medium. At 8.0 GHz, one turn of the barrel will change the phase of the signal by 5° giving the phase adjuster a range of approximately 60°.

Attenuators are also added to each signal line. The purpose of the first set of attenuators is to balance the power between the reference signal and the modulus signal at

the output of the second amplification stage. A second set of attenuators is added to reduce the power input to the mixers, since the 20 dBm output of the second stage amplifiers saturates the mixers, causing phase errors.

G. OVERALL DESIGN

The system is assembled on a 1/4" brass plate measuring 13 in. by 13in. The plate serves two purposes. The first is structural: by attaching the components to the plate any vibrational errors that might be introduced are eliminated. The second function of the brass plate is as a heat sink. The amplifiers will operate at a high temperature, which may introduce phase errors into the system. The brass plate keeps the entire system at a constant temperature that is only slightly above room temperature, thus minimizing errors. All coaxial cables are cut to length. These custom fit lines minimize bends in the cables and the number of connectors that are needed to assemble the system. Figures 5.10 through 5.12 show the front, bottom, and top of the antenna system.

Figure 5.10: Front View of RSNS Antenna.

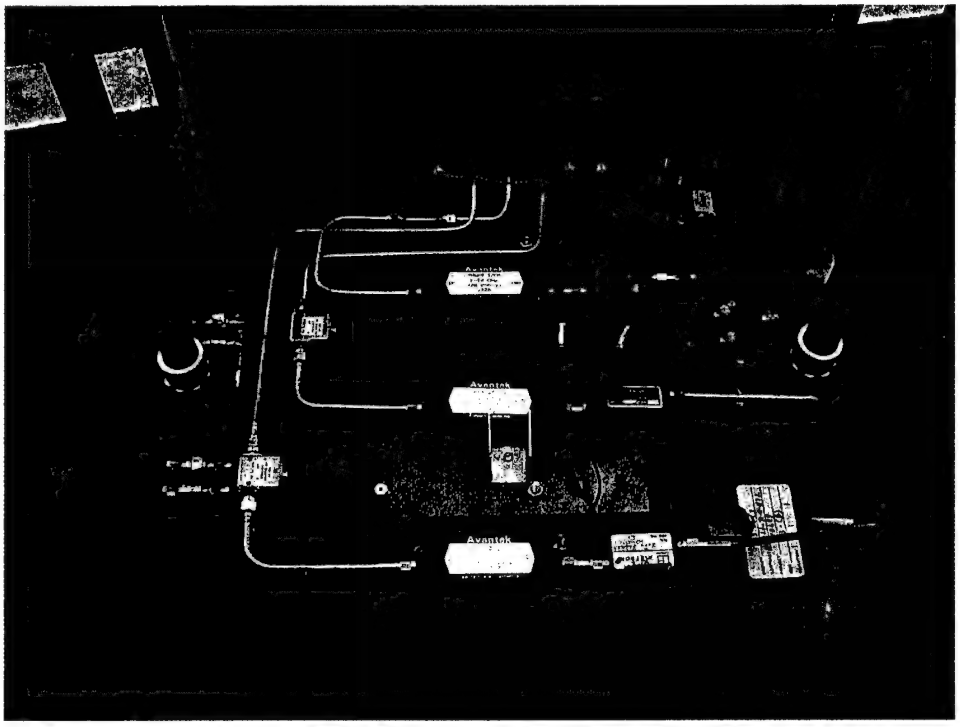


Figure 5.11: Bottom View of RSNS Antenna.

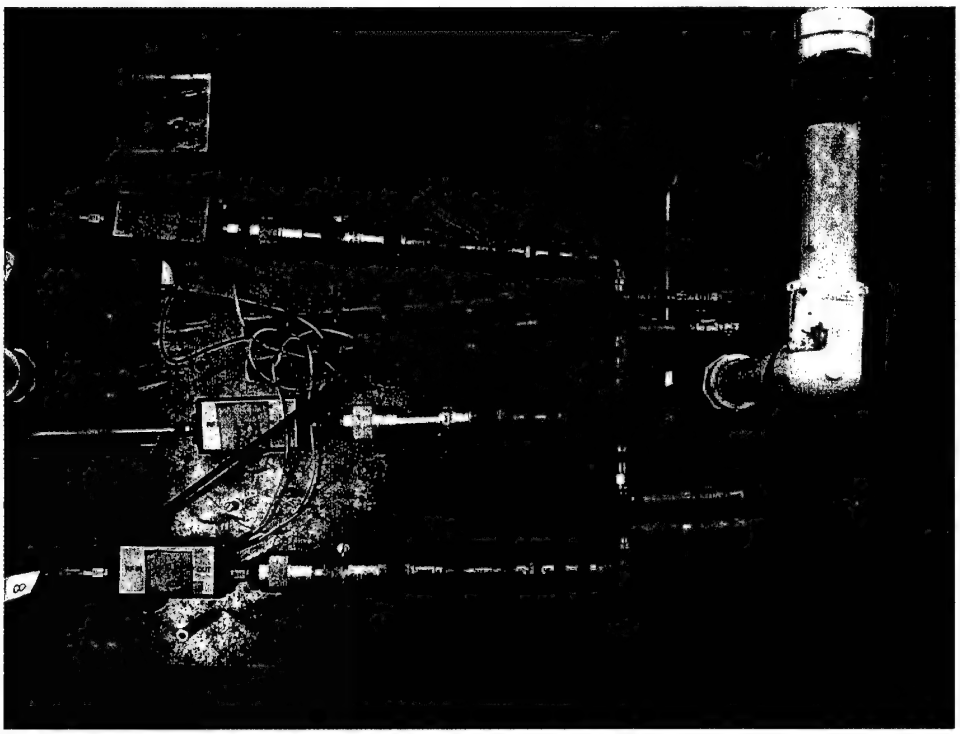


Figure 5.12: Top View of RSNS Antenna.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. RSNS SIGNAL PROCESSING CIRCUIT

The second half of the DF system is the RSNS signal processing circuit. This circuit is responsible for converting the folding waveforms into angles of arrival. The circuit consists of 4 basic parts: level and bias amplifiers, comparator network, RSNS-to-binary converters and the timing circuit. Figure 6.1 is a block diagram of the entire circuit.

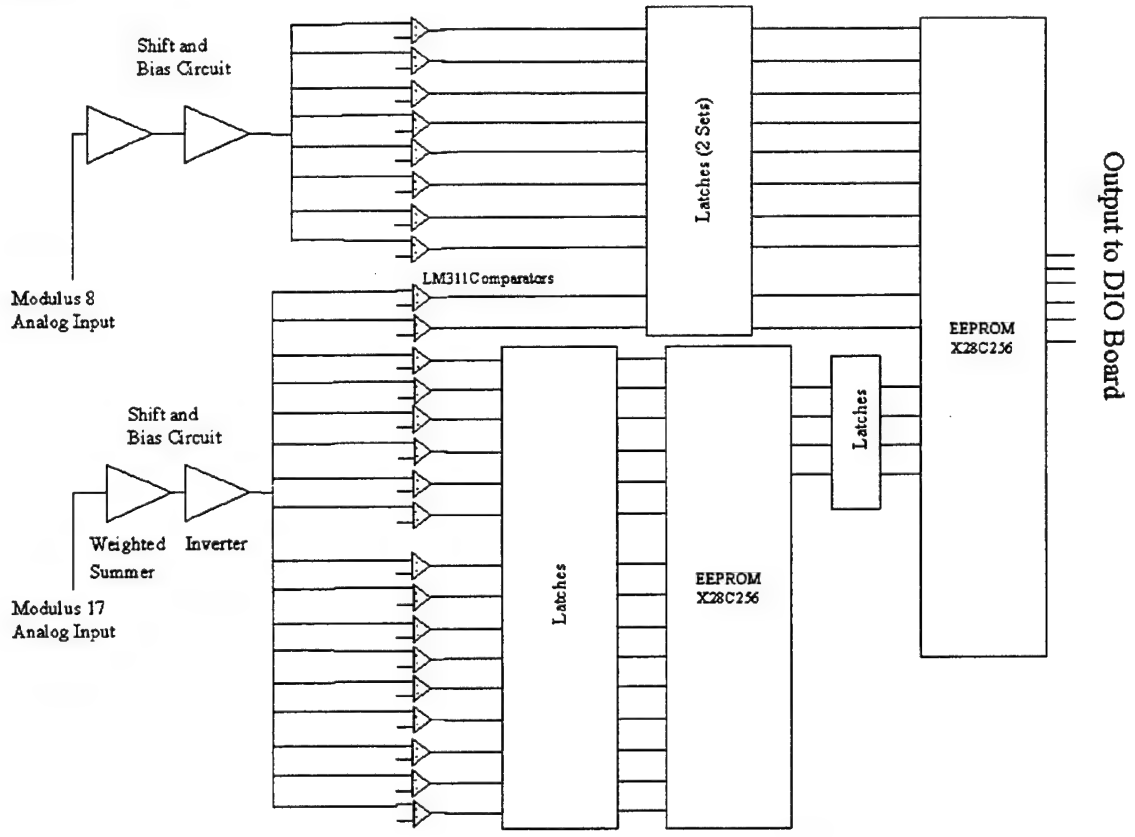


Figure 6.1: Block Diagram of RSNS Processor.

The circuit gets its input from the balanced mixers and the folding waveform is sent through the level and bias amplifiers. This part of the system converts the folding waveforms (phase response from each channel) from a ±320 millivolt signal to a 1 to 13 volt signal. The signal is then sent to a comparator network. The comparator network operates like a thermometer, the higher the voltage of the signal, the more comparators that are turned on. The next portion of the system to receive the signal is the timing

circuit. This circuit is responsible for latching the comparator outputs to the EEPROMS to perform the RSNS-to-binary conversion. The conversion is simply a matter of comparing the number of comparators that are turned on in each of the channels with the RSNS code, generating a binary number between 0 and 63 that corresponds to an angle of arrival.

The improvements made to the first prototype signal processing design include the addition of a timing circuit to latch the signal inputs to the EEPROMs (RSNS-to-binary converter) and constructing the preprocessing circuit on a printed circuit board (PCB). These modifications were added to increase the reliability of the system.

A. LEVEL AND BIAS AMPLIFIER

In order for the comparator network to function efficiently the ± 320 millivolt signal needs to be conditioned. The first item to consider is that the signal takes on both positive and negative values. This changing polarity of the signal requires that the comparators have multiple power sources. To avoid multiple power sources a 7.0 volt bias is introduced. The second item to consider is that the range of the signal is only 640 millivolts. For this case, the comparator voltage references would require very tight tolerances and very accurate comparators. This problem is alleviated by amplifying the signal so that its range is now 12 volts centered about a 7 volt bias.

A weighted summing amplifier with an inverter is used to condition the signal. Figure 6.2 is a diagram of the weighted summing amplifier. The amplifier is a National Instruments LM-741CN Operational Amplifier. This amplifier has two inputs: the mixer output (Mod Signal) and V_2 that acts as a bias voltage. R_2 determines the amount of bias V_2 provides. R_1 determines the influence generated by the mixer input. R_3 sets the gain. The equation that governs the amplifier operation is the summing amplifier gain equation:

$$V_{out} = -R_3\left(\frac{V_1}{R_1} + \frac{V_2}{R_2}\right). \tag{6.1}$$

With an +18 volt source, V_2 is 18 volts and R_3 is set to $10k\Omega$. The two remaining resistance values are determined by matching the mixer output maximum and minimum values to the desired outputs of 1-13 volts. The negative sign on the right side of Equation (6.1) indicates that this is an inverting amplifier. The mixer output has a DC bias of -40 millivolts; however, when a 7 volt bias is introduced it becomes insignificant and is ignored. A phase detector output of zero maps to the middle of the shift and bias band: -7 volts. When V_1=0, (6.1) reduces to

$$R_2 = -R_3\frac{V_2}{V_{out}} = -10k\Omega \cdot \frac{18V}{-7V} = 25.7k\Omega. \tag{6.2}$$

The value of R_1 is determined from the maximum value of the mixer output. For the modulus 8 and 17 channels, the maximum mixer output is measured as 320 millivolts and the output is –13 volts. R_1 is determined as follows:

$$R_1 = \frac{-V_1}{\left(\frac{V_{out}}{R_3} + \frac{V_2}{R_2}\right)} = \frac{-0.320V}{\left(\frac{-13V}{10k\Omega} + \frac{18V}{25.7k\Omega}\right)} = \frac{-0.320V}{-600\mu A} = 533\Omega. \tag{6.3}$$

Here R_1 is a 1 $k\Omega$ variable resistor set to 533 Ω and R_2 is a 24 $k\Omega$ resistor in series with a 1.6 $k\Omega$ resistor.

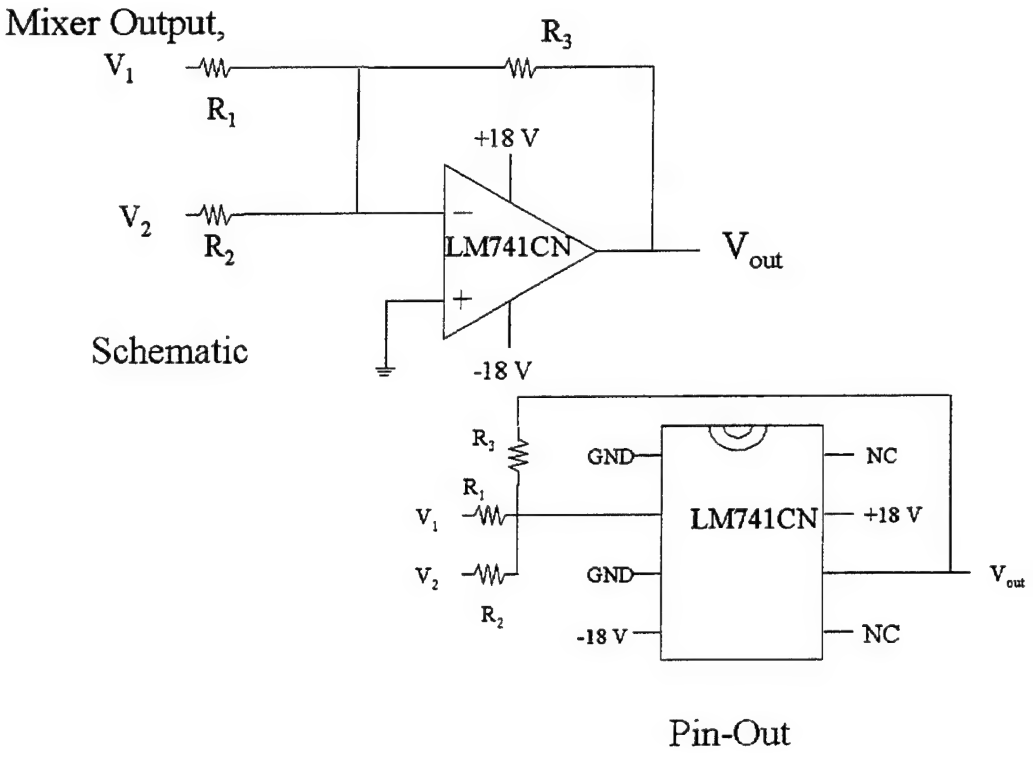


Figure 6.2: Weighted Summing Amplifier [from 5].

The output of the weighted summing amplifier must be inverted before transmission to the comparators. The inverter shown in Figure 6.3 performs this. No amplification is required in the inverting stage because the gain of the summing amplifier is correct . The equation that governs the output of an inverting amplifier is

$$V_{out} = -\frac{R_1}{R_2} V_{in} . \tag{6.4}$$

To obtain unity gain with inversion, R_1=R_2, 20kΩ resistors are used. The output of the inverter is +1 to +13 volts as desired. This signal is provided to the comparator network to determine the level of each signal [5].

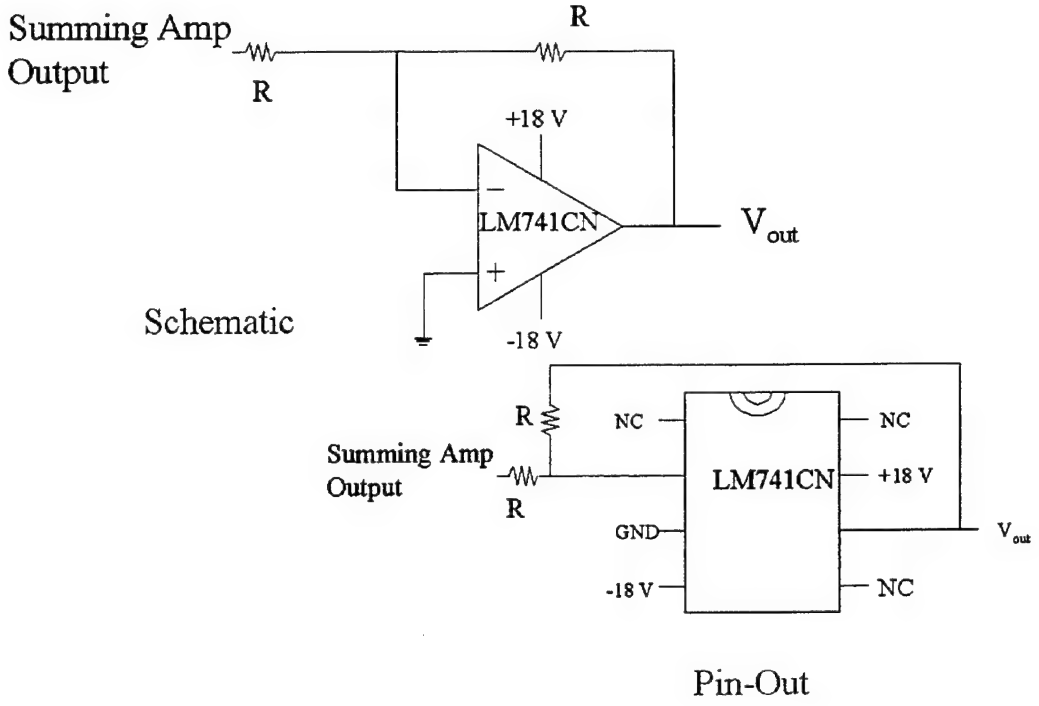


Figure 6. 3: Inverting Amplifier [from 5].

B. COMPARATOR NETWORK

The comparator networks have 8 and 17 comparators in parallel. The circuit for each comparator is shown in Figure 6.4.

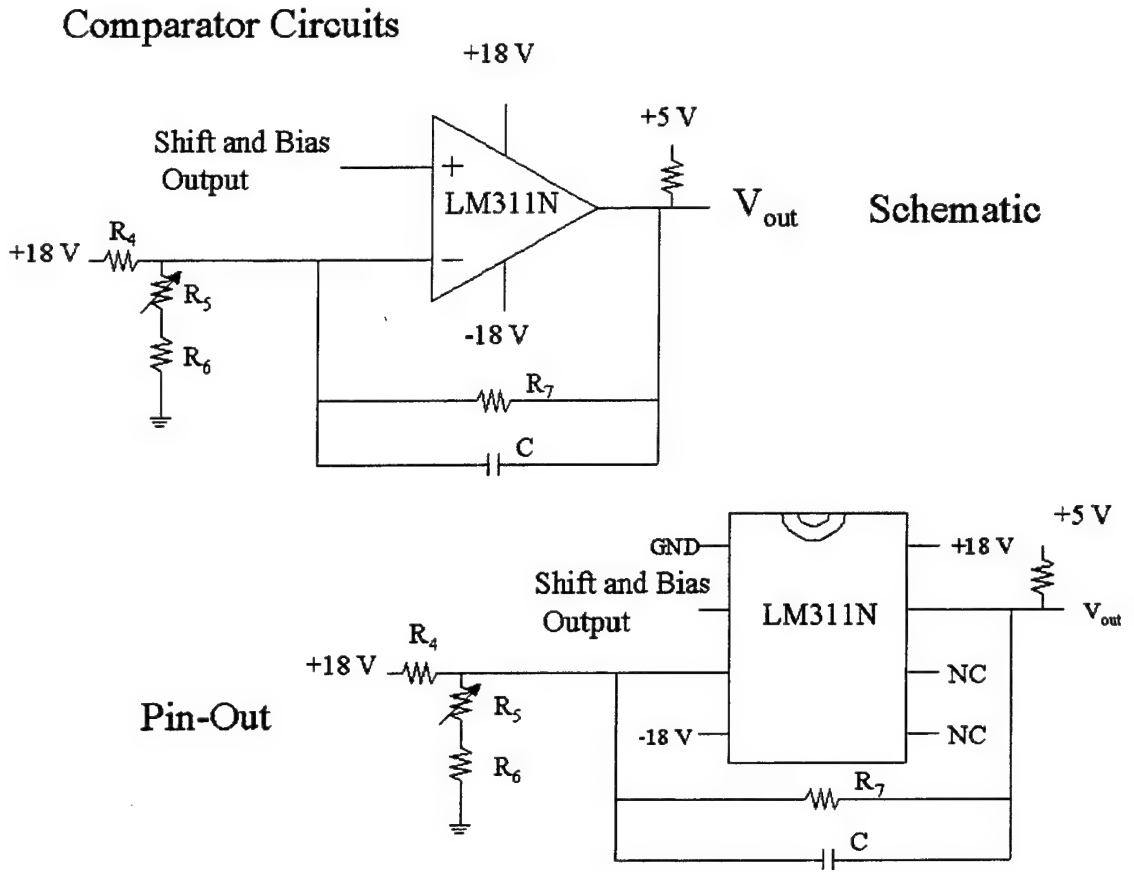


Figure 6.4: Comparator Schematic and Pin-out Diagram.

The shift and bias circuit output is the input to the non-inverting terminal. The second input is a reference voltage generated in the voltage divider network shown in Figure 6.5. Since the 18-volt power supply is available as the power supply for the comparator, it is used as the source for the voltage divider. The thresholds for the comparators, as explained in the previous section, range from 1 to 13 volts, so the conversion from the normalized thresholds to the actual thresholds is $V_{act} = 6V_{norm} + 7$. The normalized and actual threshold values are shown in columns two and three of Table 6.1. Applying Ohm's law between the source and inverting terminal of the comparator, Equation 6.4 shows the relationship between I_2 and R_4:

$$I_2 = \frac{18V - V_{threshold}}{R_5}. \tag{6.4}$$

The comparators (National Semiconductor LM311N) have a maximum input bias and offset current of 100 nA. This current is I_2 in Figure 6.5. In order to limit the input offset and bias currents to less than 1% of the current in the voltage divider, the total resistance of the network must not exceed $1.5M\Omega$. However, the resistance values must be large enough such that the current required by the system remains small. For these reasons, R_4 is chosen to be $100k\Omega$. The total current for the voltage dividers is 27.5mA as shown in the lower left corner of Table 6.1. Kirchhoff's Current Law applied at the node above R_2 yields $I_3 = I_1 - I_2$. The only variables that remain are R_5 and R_6. Ohm's law across these resistors dictates:

$$R_4 + R_5 = \frac{V_{thresh}}{I_3}. \tag{6.5}$$

As an initial value, R_5 is chosen to be 5% of R_6. Once initial values are determined, standard resistor values are chosen and R_5 is made a variable resistor. The voltage divider sub-circuit is complete and provides a precise threshold voltage to the inverting terminal of the comparator [5].

Simulation Thresholds — Given R4=100k and I2=100 nA

Mod 8 Channel — Resistance Values

	Normalized	Comparator	R4		R5		Use Var Res	R6		Use Resistor
t1=	-0.98079	1.11528	811	1.000E+05	812	314.723	1K	813	6294.455	5.6K
t2=	-0.83147	2.01116	821	1.000E+05	822	5.994E+02	1K	823	1.199E+04	12K
t3=	-0.55557	3.66855	831	1.000E+05	832	1.219E+03	5K	833	2.438E+04	22K
t4=	-0.19509	5.82943	841	1.000E+05	842	2.283E+03	2K	843	4.565E+04	47K
t5=	0.19509	8.17052	851	1.000E+05	852	3.962E+03	10K	853	79245.061	75K
t6=	0.55557	10.33341	861	1.000E+05	862	6.427E+03	20K	863	1.285E+05	120K
t7=	0.83147	11.98881	871	1.000E+05	872	9.513E+03	20K	873	1.903E+05	180K
t8=	0.98079	12.88471	881	1.000E+05	882	1.202E+04	50K	883	2.404E+05	220K
Mod 17 Channel										
t1=	-0.99573	1.02559	711	1.000E+05	712	2.879E+02	1K	713	5.758E+03	5.6K
t2=	-0.96183	1.22905	721	1.000E+05	722	3.492E+02	1K	723	6.984E+03	6.8K
t3=	-0.89516	1.62902	731	1.000E+05	732	4.741E+02	2K	733	9.483E+03	9.1K
t4=	-0.79802	2.21190	741	1.000E+05	742	6.676E+02	5K	743	1.335E+04	12K
t5=	-0.67370	2.95782	751	1.000E+05	752	9.370E+02	5K	753	1.874E+04	18K
t6=	-0.52643	3.84141	761	1.000E+05	762	1.293E+03	10K	763	2.586E+04	22K
t7=	-0.36124	4.83255	771	1.000E+05	772	1.749E+03	10K	773	3.498E+04	33K
t8=	-0.18375	5.89750	781	1.000E+05	782	2.322E+03	5K	783	4.645E+04	47K
t9=	0.00000	7.00000	791	1.000E+05	792	3.033E+03	10K	793	6.066E+04	56K
t10=	0.18375	8.10250	7A1	1.000E+05	7A2	3.902E+03	10K	7A3	7.804E+04	75K
t11=	0.36124	9.16745	7B1	1.000E+05	7B2	4.948E+03	5K	7B3	9.896E+04	100K
t12=	0.52643	10.15859	7C1	1.000E+05	7C2	6.177E+03	20K	7C3	1.235E+05	120K
t13=	0.67370	11.04217	7D1	1.000E+05	7D2	7.568E+03	20K	7D3	1.514E+05	150K
t14=	0.79802	11.78810	7E1	1.000E+05	7E2	9.051E+03	20K	7E3	1.810E+05	150K
t15=	0.89516	12.37098	7F1	1.000E+05	7F2	1.048E+04	50K	7F3	2.097E+05	200K
t16=	0.96183	12.77095	7G1	1.000E+05	7G2	1.165E+04	50K	7G3	2.330E+05	300K
t17=	0.99573	12.97441	7H1	1.000E+05	7H2	1.232E+04	50K	7H3	2.464E+05	300K

Current Values — Input Bias Current 1.00E-07

I1		I2		I3		
811	1.688E-04	812	1.00E-07	813	1.687E-04	
821	1.599E-04	822	1.00E-07	823	1.598E-04	
831	1.433E-04	832	1.00E-07	833	1.432E-04	
841	1.217E-04	842	1.00E-07	843	1.216E-04	
851	9.829E-05	852	1.00E-07	853	9.819E-05	
861	7.667E-05	862	1.00E-07	863	7.657E-05	
871	6.011E-05	872	1.00E-07	873	6.001E-05	
881	5.115E-05	882	1.00E-07	883	5.105E-05	
711	1.697E-04	712	1.00E-07	713	1.696E-04	
721	1.677E-04	722	1.00E-07	723	1.676E-04	
731	1.637E-04	732	1.00E-07	733	1.636E-04	
741	1.579E-04	742	1.00E-07	743	1.578E-04	
751	1.504E-04	752	1.00E-07	753	1.503E-04	
761	1.416E-04	762	1.00E-07	763	1.415E-04	
771	1.317E-04	772	1.00E-07	773	1.316E-04	
781	1.210E-04	782	1.00E-07	783	1.209E-04	
791	1.100E-04	792	1.00E-07	793	1.099E-04	
7A1	9.898E-05	7A2	1.00E-07	7A3	9.888E-05	A=10
7B1	8.833E-05	7B2	1.00E-07	7B3	8.823E-05	B=11
7C1	7.841E-05	7C2	1.00E-07	7C3	7.831E-05	C=12
7D1	6.958E-05	7D2	1.00E-07	7D3	6.948E-05	D=13
7E1	6.212E-05	7E2	1.00E-07	7E3	6.202E-05	E=14
7F1	5.629E-05	7F2	1.00E-07	7F3	5.619E-05	F=15
7G1	5.229E-05	7G2	1.00E-07	7G3	5.219E-05	G=16
7H1	5.026E-05	7H2	1.00E-07	7H3	5.016E-05	H=17

Table 6.1: Voltage Divider Calculations.

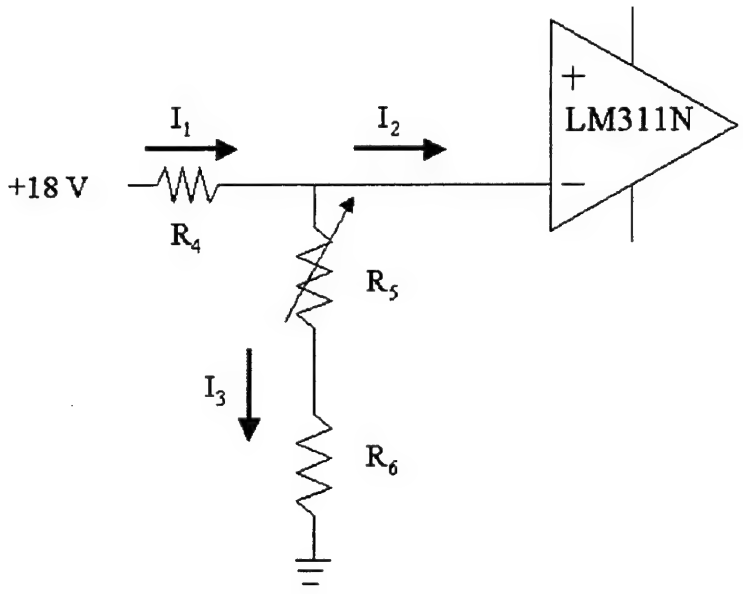


Figure 6.5: Voltage Divider Circuit [from 5].

To improve the comparator's performance a Schmitt trigger is added. R_7 is introduced to add hysteresis to the comparator characteristics. This gives the comparator two threshold values: V_{TL} and V_{TH}. Figure 6.6 shows the transfer characteristic of the comparator circuit. V_{thresh} is set by the voltage divider and L is the comparator output signal. V_{TL} and V_{TH} can be found by equations

$$V_{TL} = V_{thresh} - L_{+}\left(\frac{R_4 + R_5 + R_6 \Big/ R_4(R_5 + R_6)}{R_7}\right) \tag{6.6}$$

and

$$V_{TH} = V_{thresh} - L_{-}\left(\frac{R_4 + R_5 + R_6 \Big/ R_4(R_5 + R_6)}{R_7}\right). \tag{6.7}$$

The power supply of the LM311 comparator sets the output signal, $L=V_{CC}$, in this case the $L_{+} = 18$ volts and $L_{-} = -18$ volts.

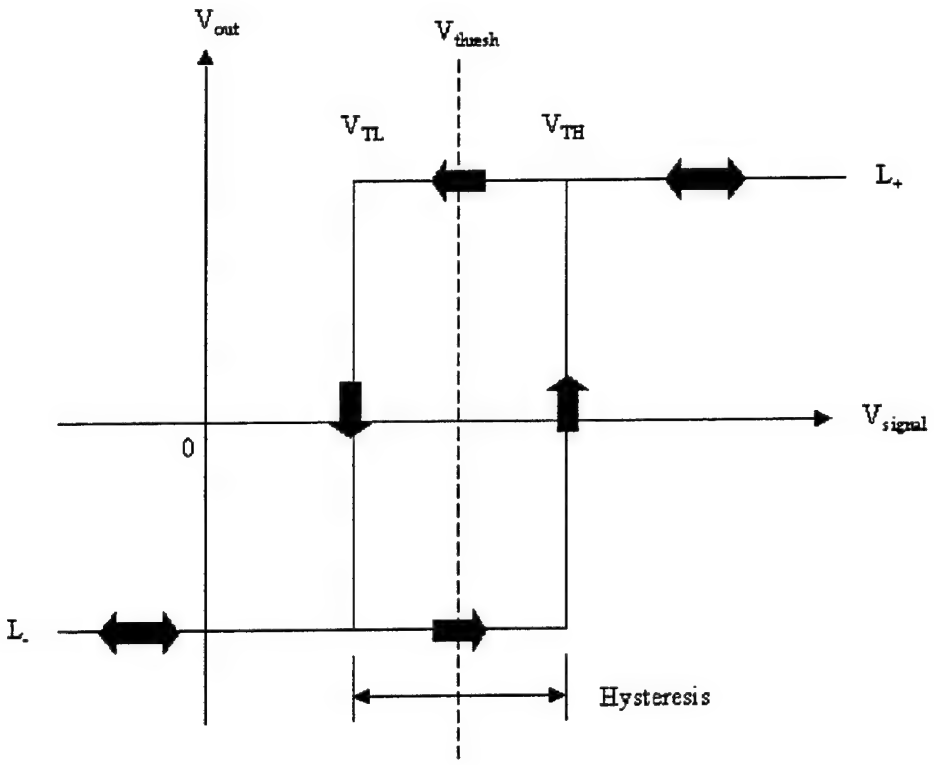


Figure 6.6: Comparator Characteristic with Hysteresis.

A small capacitor, C, is added across the feedback resistor to ensure fast transitions and to prevent multiple pulses as the signal passes though the threshold voltages. The value of 10 MΩ is selected for R_4 and 100 pF for C [11, 12]. Table 6.2 lists the threshold values for each comparator.

L = 18 V R7 = 10M

Comparator	V_{TL}	V_{TH}
1.11528	1.11528	1.11528
2.01116	2.01116	2.01116
3.66655	3.66655	3.66655
5.82943	5.82943	5.82943
8.17052	8.17052	8.17052
10.33341	10.33341	10.33341
11.98881	11.98881	11.98881
12.88471	12.88471	12.88471
1.02559	1.02559	1.02559
1.22905	1.22905	1.22905
1.62902	1.62902	1.62902
2.21190	2.21190	2.21190
2.95782	2.95782	2.95782
3.84141	3.84141	3.84141
4.83255	4.83255	4.83255
5.89750	5.89750	5.89750
7.00000	7.00000	7.00000
8.10250	8.10250	8.10250
9.16745	9.16745	9.16745
10.15859	10.15859	10.15859
11.04217	11.04217	11.04217
11.78810	11.78810	11.78810
12.37098	12.37098	12.37098
12.77095	12.77095	12.77095
12.97441	12.97441	12.97441

Table 6.2: Comparator Threshold Settings.

When the shift and bias signal is larger than V_{TH}, the comparator is ON and remains on until the signal falls below V_{TL}. In ON state, no current is accepted at the output, so the 5 volt signal above the output resistor has no voltage drop. The 5 volts is impressed on the output. This voltage is passed to the input of the latch. If the reference voltage exceeds the shift and bias signal, the comparator is OFF. In this state, the output sinks current, and the 5 volts are dropped across the output resistor. The output of the comparator is shorted to ground through the opamp and passed to the input of the latch.

C. LATCHING THE COMPARATOR OUTPUTS

A D flip-flop is added to the prototype design to latch the data to the EEPROMs. A set of latches is placed on the inputs for each of the two EEPROMs. A total of 39 latches are required for the design. Six 74HC273N octal D type flip-flops are used as latches. Figure 6.7 shows the pinout for the 74HC273 chips.

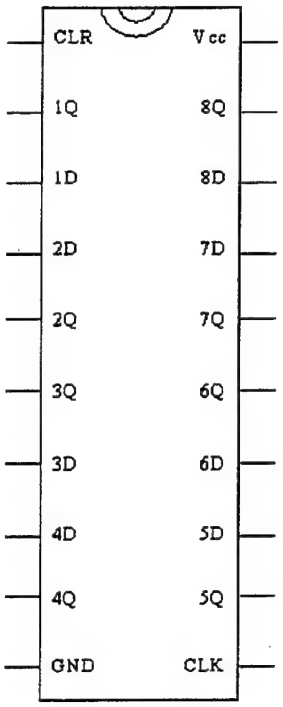


Figure 6.7: 74HC273N Octal D-type Flip-flop.

The V_{cc} and CLR pins are set to 5 volts, D pins are the inputs, Q pins are the outputs and the CLK pin receives a signal from a 555 astable multivibrator. This pinout simply stores the input value on the D pin for one clock cycle. By latching the value for one clock cycle the inputs to the EEPROMs are stabilized.

The clock signal is generated by 555 timer circuit shown in Figure 6.8. R_A, R_B, and C control the period and duty cycle. The design equations governing this circuit are

$$\text{Period} = 0.69C(R_A + 2R_B) \tag{6.8}$$

and

$$\text{Duty Cycle} = \frac{R_A + R_B}{R_A + 2R_B}. \tag{6.9}$$

To select a clock signal of 1 kHz and a duty cycle of 50%, R_A, R_B, and C are set to 1.3 kΩ, 6.2 kΩ, and 100 nF respectively [12].

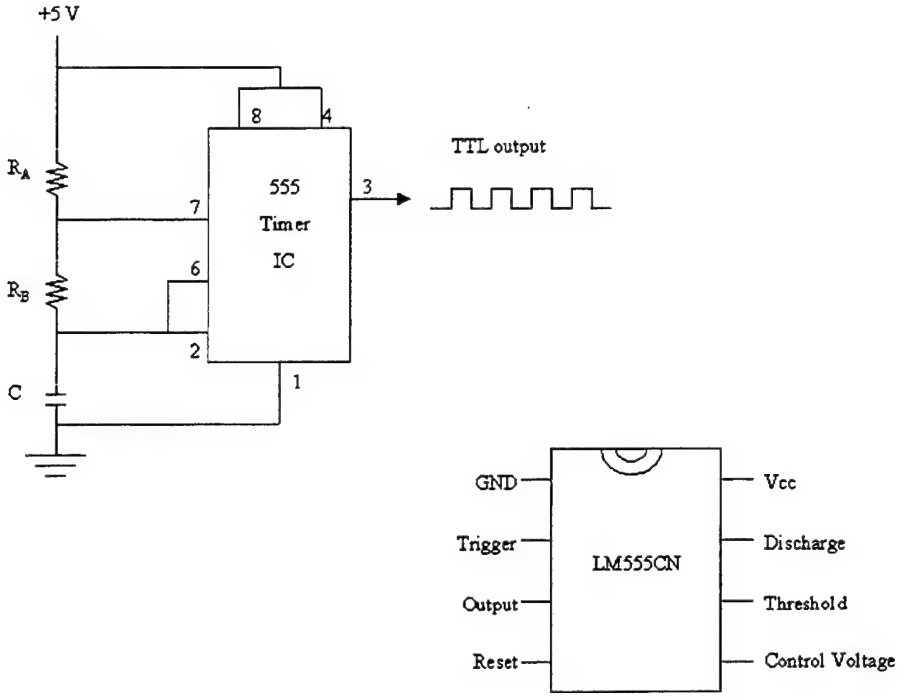


Figure 6.8: 555 Timer Circuit.

D. RSNS-TO-BINARY CONVERTER

The 25 comparator states must be processed into a 6-bit angle of arrival. The EEPROM performs this mapping function. An ATMEL Model AT28C256 is used. The EEPROM is a small logic block that converts 15 input bits to an 8-bit output via its internal programming. Because of the 15 input limit, two EEPROM are used to process all 25 comparators. The "Thermometer Code" programmed into the EEPROM is shown in Table 6.3. The thermometer code is the string of instructions for the EEPROM that maps input to output. With two EEPROMs, the thermometer code has two stages as shown on the circuit diagrams of Figure 6.9 and Figure 6.10.

RSNS Encoding Scheme

			XICOR #1:10 LSB of Mod 17 Channel				XICOR #2:7 MSB of Mod 17 Channel			
Bin	Channel 17	Channel 8	Input XICOR #1	Hex Input XICOR #1	Output XICOR #1	Hex Out XICOR #1	Input XICOR#2	Hex Input XICOR#2	Output XICOR#2	Hex Out XICOR#2
0	10	6	000 0011 1111 1111	03FF	10	A	000 1111 1100 1010	0FCA	0	00
1	10	5	000 0011 1111 1111	03FF	10	A	000 0111 1100 1010	07CA	1	01
2	11	5	000 0111 1111 1111	07FF	11	B	000 0111 1100 1011	07CB	2	02
3	11	4	000 0111 1111 1111	07FF	11	B	000 0011 1100 1011	03CB	3	03
4	12	4	000 1111 1111 1111	0FFF	12	C	000 0011 1100 1100	03CC	4	04
5	12	3	000 1111 1111 1111	0FFF	12	C	000 0001 1100 1100	01CC	5	05
6	13	3	001 1111 1111 1111	1FFF	13	D	000 0001 1100 1101	01CD	6	06
7	13	2	001 1111 1111 1111	1FFF	13	D	000 0000 1100 1101	00CD	7	07
8	14	2	011 1111 1111 1111	3FFF	14	E	000 0000 1100 1110	00CE	8	08
9	14	1	011 1111 1111 1111	3FFF	14	E	000 0000 0100 1110	004E	9	09
10	15	1	111 1111 1111 1111	7FFF	15	F	000 0000 0100 1111	004F	10	0A
11	15	0	111 1111 1111 1111	7FFF	15	F	000 0000 0000 1111	000F	11	0B
12	16	0	111 1111 1111 1111	7FFF	15	F	000 0000 0001 1111	001F	12	0C
13	16	1	111 1111 1111 1111	7FFF	15	F	000 0000 0101 1111	005F	13	0D
14	17	1	111 1111 1111 1111	7FFF	15	F	000 0000 0111 1111	007F	14	0E
15	17	2	111 1111 1111 1111	7FFF	15	F	000 0000 1111 1111	00FF	15	0F
16	16	2	111 1111 1111 1111	7FFF	15	F	000 0000 1101 1111	00DF	16	10
17	16	3	111 1111 1111 1111	7FFF	15	F	000 0001 1101 1111	01DF	17	11
18	15	3	111 1111 1111 1111	7FFF	15	F	000 0001 1100 1111	01CF	18	12
19	15	4	111 1111 1111 1111	7FFF	15	F	000 0011 1100 1111	03CF	19	13
20	14	4	011 1111 1111 1111	3FFF	14	E	000 0011 1100 1110	03CE	20	14
21	14	5	011 1111 1111 1111	3FFF	14	E	000 0111 1100 1110	07CE	21	15
22	13	5	001 1111 1111 1111	1FFF	13	D	000 0111 1100 1101	07CD	22	16
23	13	6	001 1111 1111 1111	1FFF	13	D	000 1111 1100 1101	0FCD	23	17
24	12	6	000 1111 1111 1111	0FFF	12	C	000 1111 1100 1100	0FCC	24	18
25	12	7	000 1111 1111 1111	0FFF	12	C	001 1111 1100 1100	1FCC	25	19
26	11	7	000 0111 1111 1111	07FF	11	B	001 1111 1100 1011	1FCB	26	1A
27	11	8	000 0111 1111 1111	07FF	11	B	011 1111 1100 1011	3FCB	27	1B
28	10	8	000 0011 1111 1111	03FF	10	A	011 1111 1100 1010	3FCA	28	1C
29	10	7	000 0011 1111 1111	03FF	10	A	001 1111 1100 1010	1FCA	29	1D
30	9	7	000 0001 1111 1111	01FF	9	9	001 1111 1100 1001	1FC9	30	1E
31	9	6	000 0001 1111 1111	01FF	9	9	000 1111 1100 1001	0FC9	31	1F
32	8	6	000 0000 1111 1111	00FF	8	8	000 1111 1100 1000	0FC8	32	20
33	8	5	000 0000 1111 1111	00FF	8	8	000 0111 1100 1000	07C8	33	21
34	7	5	000 0000 0111 1111	007F	7	7	000 0111 1100 0111	07C7	34	22
35	7	4	000 0000 0111 1111	007F	7	7	000 0011 1100 0111	03C7	35	23
36	6	4	000 0000 0011 1111	003F	6	6	000 0011 1100 0110	03C6	36	24
37	6	3	000 0000 0011 1111	003F	6	6	000 0001 1100 0110	01C6	37	25
38	5	3	000 0000 0001 1111	001F	5	5	000 0001 1100 0101	01C5	38	26
39	5	2	000 0000 0001 1111	001F	5	5	000 0000 1100 0101	00C5	39	27
40	4	2	000 0000 0000 1111	000F	4	4	000 0000 1100 0100	00C4	40	28
41	4	1	000 0000 0000 1111	000F	4	4	000 0000 0100 0100	0044	41	29
42	3	1	000 0000 0000 0111	0007	3	3	000 0000 0100 0011	0043	42	2A
43	3	0	000 0000 0000 0111	0007	3	3	000 0000 0000 0011	0003	43	2B
44	2	0	000 0000 0000 0011	0003	2	2	000 0000 0000 0010	0002	44	2C
45	2	1	000 0000 0000 0011	0003	2	2	000 0000 0100 0010	0042	45	2D
46	1	1	000 0000 0000 0001	0001	1	1	000 0000 0100 0001	0041	46	2E
47	1	2	000 0000 0000 0001	0001	1	1	000 0000 1100 0001	00C1	47	2F
48	0	2	000 0000 0000 0000	0000	0	0	000 0000 1100 0000	00C0	48	30
49	0	3	000 0000 0000 0000	0000	0	0	000 0001 1100 0000	01C0	49	31
50	1	3	000 0000 0000 0001	0001	1	1	000 0001 1100 0001	01C1	50	32
51	1	4	000 0000 0000 0001	0001	1	1	000 0011 1100 0001	03C1	51	33
52	2	4	000 0000 0000 0011	0003	2	2	000 0011 1100 0010	03C2	52	34
53	2	5	000 0000 0000 0011	0003	2	2	000 0111 1100 0010	07C2	53	35
54	3	5	000 0000 0000 0111	0007	3	3	000 0111 1100 0011	07C3	54	36
55	3	6	000 0000 0000 0111	0007	3	3	000 1111 1100 0011	0FC3	55	37
56	4	6	000 0000 0000 1111	000F	4	4	000 1111 1100 0100	0FC4	56	38
57	4	7	000 0000 0000 1111	000F	4	4	001 1111 1100 0100	1FC4	57	39
58	5	7	000 0000 0001 1111	001F	5	5	001 1111 1100 0101	1FC5	58	3A
59	5	8	000 0000 0001 1111	001F	5	5	011 1111 1100 0101	3FC5	59	3B
60	6	8	000 0000 0011 1111	003F	6	6	011 1111 1100 0110	3FC6	60	3C
61	6	7	000 0000 0011 1111	003F	6	6	001 1111 1100 0110	1FC6	61	3D
62	7	7	000 0000 0111 1111	007F	7	7	001 1111 1100 0111	1FC7	62	3E
63	7	6	000 0000 0111 1111	007F	7	7	000 1111 1100 0111	0FC7	63	3F

Table 6.3: RSNS Encoding Scheme [from 5].

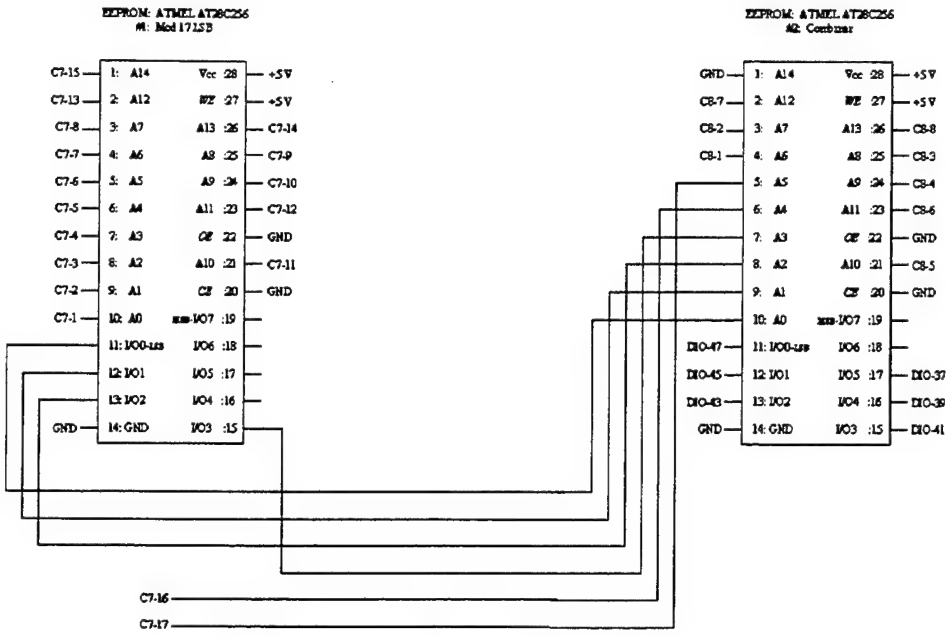


Figure 6.9: Schematic Diagram of Mod 17 Comparator Output to EEPROM#2 Input.

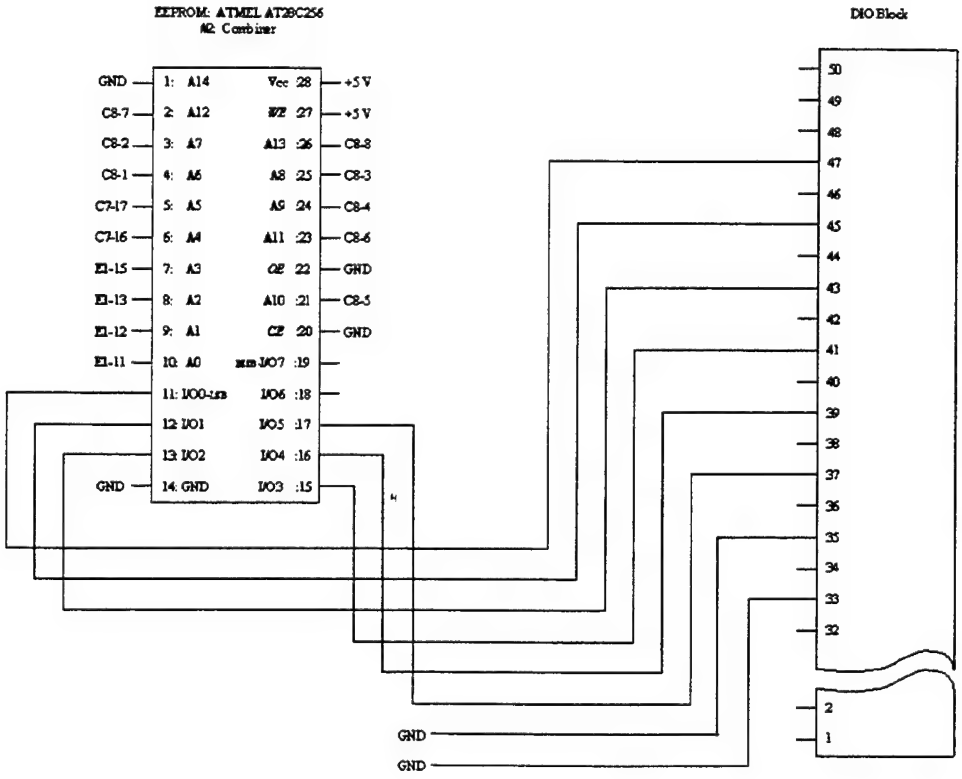


Figure 6.10: Schematic Diagram of Mod 8 Comparator to DIO Board Wiring.

The first stage encodes the 15 least significant (lowest threshold) comparators of the modulus 17 channel into a 4-bit representation (thermometer-to-binary conversion). The first column of Table 6.2 is the reported angle bin. The second column is the number of comparators active in the modulus 17 channel in each bin. The fourth column is a binary representation of the state of the input pins for the first EEPROM, and the fifth column is the hexadecimal representation of that binary number. The sixth column is the decimal output desired from EEPROM #1 for the given input; it is the decimal number of number of active comparators. The seventh column is the hexadecimal representation of that digital number. The result is that the number of active comparators is converted to a 4-bit number. The shift and bias circuit and comparators for Modulus 17 and EEPROM #1 are physically located on the same circuit board. The output of EEPROM #1 and comparators 16 and 17 must be passed to EEPROM #2 on the second circuit board. Figure 6.9 illustrates how this signal is passed to EEPROM #2 [5].

E. CIRCUIT SIMULATION

A schematic capture and simulation software package is used to test and validate the entire circuit design. Multisim ™ from Electronic Workbench was chosen because it is powerful, easy to learn and use. No knowledge of SPICE or other simulation language is needed to conduct basic simulations. The only knowledge that is necessary is basic circuitry design. The interface is graphical and very intuitive.

Validation for each of the parts of the RSNS processor is done by selecting all the individual resistors, capacitors, integrated circuits, power supplies, and connecting the correct pins together. The program allows the user to add various measuring equipment to the circuit to monitor performance. Items such as oscilloscopes and voltmeters are

very useful to verify and tweak the initial design. The built-in function generator was used to provide the mixer output signal and each of the EEPROM inputs was monitored to ensure that the comparator network operated properly.

A simulation of the complete system was not completed. The difficulty arose from the EEPROM. No standard SPICE model exists for EEPROMs, so a model needed to be created. Included in Multisim™ is a component creator that allows the user to generate their own unique components. This ability is limited to basic components such as diodes and opamps. In order to create a model for the EEPROMs a VHSIC Hardware Description Language (VHDL) add-on package is required. This language allows a programmer to model all levels of circuit design. It supports behavioral description of hardware from the digital system level to the gate level. The language is able to take a black-box approach to designing a system since all that is required in input/output information. This makes the language technology independent and extremely flexible.

Three approaches were considered to simulate the behavior of the EEPROM. The first method is the easiest conceptually and most direct method. The truth table is entered into the VHDL code as a constant [13]. For smaller truth tables this method gives the programmer direct control of all input and output bits. For this system, there are 15 input bits that need to be accounted for thus requiring a truth table that has 2^{15} lines. The second method uses a programmable logic array (PLA) instead of truth tables. The table is replaced with an AND array that realizes selected product terms of the input variables and an OR array that operates on the product terms to form an output. This approach eliminates the need to include every possible input combination, thus reducing the number of lines of code. The drawback to this method though is that equations need to be developed and entered into Karnaugh maps, and a separate VHDL library file needs to be created to use this method. The third approach is the Case Model [14]. This approach is usually used to realize a multiplexer, however, it can be manipulated to simulate the operation of a ROM. The case statement allows the programmer to directly read the input bits and convert them into an output. This is similar to programming in a truth table except that only the required lines from the table are included instead of all the lines. Appendix A is the code developed to simulate the EEPROM.

Once the VHDL code was developed, it can be imported to Multisim™ and attached to a component. The component can then be added to the circuit for simulation. When the EEPROM components were added to the simulation, however, the Multisim™ package crashed. After troubleshooting, it was discovered that the VHDL interface for Multisim™ was not functioning. Electronics Workbench is developing a newer version of their design software to correct this bug, however, the fix was not available in time for the submission of this work.

F. PRINTED CIRCUIT BOARD CONSTRUCTION

In order to produce a PCB for the system, the entire circuit needs to be constructed in a program capable of generating a Gerber file (drawing file used by PCB milling machines). The Multisim/Ultiboard/Ultiroute™ software package produced by Electronics Workbench was selected to do this design. Multisim™ is a schematic capture and simulation program. It is used to design and test circuits. Once a design is complete, the program will generate a parts list and netlist, which is necessary to lay out the PCB. The Ultiboard program converts the parts list into a nearly complete scale drawing of all the components and the board itself. A board size is selected and the components are arranged to fit. Once the component layout is complete, the Ultiroute program converts the netlist into a network of traces. It does this conversion automatically and optimizes the design to reduce the number of traces required in the system. For complicated boards the routing process will invariably miss a couple of pins. This is corrected by manually adding the traces.

The next step in the process is to generate a Gerber file for production. There are a few different types Gerber files that Ultiboard™ will generate. The equipment that is used to manufacture the PCB dictates the type that needs to be generated. Once the Gerber file is created, it is sent to a manufacturing site. The PCBs that were constructed for this system were milled by the NPS Physics department. The department owns a small milling machine capable of manufacturing 11 in. by 9 in. circuit boards. Because

of the size limitations of the milling machine, two circuit boards were designed. Once the boards were milled, ground and continuity checks were completed. When each of the boards was certified, they were stuffed and soldered. Tables 6.4 and 6.5 list all the materials required to assemble the printed circuit boards.

Bill of Materials Modulus 8 PCB

Quantity	Description	Reference ID	Package
1	CONNECTORS, HDR2X7	CON1	HDR2X7
1	CONNECTORS, RIBBON_14V_MNT	CON26	RIBBON14VM
3	74LS, 74LS273N	U36, U34, U30	NO20
8	CAPACITOR, 100pF	C83, C82, C81, C80, C79, C78, C77, C76	cap2
16	RESISTOR, 100kohm	R137, R136, R135, R134, R133, R132, R131, R130, R109, R105, R101, R97, R93, R89, R85, R81	RES0.5
1	SOCKETS, DIP28	U26	DIP-28
3	POTENTIOMETER, 50K_LIN	R110, R106, R103	LIN_POT
1	RESISTOR, 330kohm	R112	RES0.5
1	RESISTOR, 240kohm	R107	RES0.5
1	RESISTOR, 120kohm	R104	RES0.5
1	RESISTOR, 75kohm	R100	RES0.5
1	POTENTIOMETER, 20K_LIN	R98	LIN_POT
1	RESISTOR, 47kohm	R95	RES0.5
2	POTENTIOMETER, 10K_LIN	R94, R91	LIN_POT
9	RESISTOR, 22kohm	R92, R111, R108, R102, R99, R96, R90, R87, R84	RES0.5
1	RESISTOR, 12kohm	R88	RES0.5
1	RESISTOR, 6.2kohm	R83	RES0.5
20	CAPACITOR, 100nF	C58, C57, C56, C55, C54, C53, C52, C51, C50, C49, C48, C47, C46, C45, C44, C43, C42, C41, C40, C39	CAP4
8	COMPARATOR, LM311N	U22, U20, U18, U16, U14, U12, U10, U8	DIP-8(N08E)
3	POTENTIOMETER, 1K_LIN	R86, R82, R79	LIN_POT
2	RESISTOR, 20kohm	R80, R75	RES0.5
1	RESISTOR, 1.6ohm	R78	RES0.5
1	RESISTOR, 24kohm	R77	RES0.5
1	RESISTOR, 10kohm	R76	RES0.5
2	OPAMP, LM741CN	U6, U4	DIP-8(N08E)

Table 6.4: Bill of Materials for Modulus 8 PCB.

Bill of Materials Modulus 17 PCB

Quantity	Description	Reference_ID	Package
1	RESISTOR, 6.2kohm	R139	RES0.5
1	RESISTOR, 1.3kohm	R138	RES0.5
17	RESISTOR, 10Mohm	R129, R124, R123, R128, R127, R122, R121, R126, R125, R120, R119, R118, R117, R116, R115, R114, R113	RES0.5
1	CONNECTORS, HDR2X7	CON18	HDR2X7
1	TIMER, LM555CN	U38	N08E
39	CAPACITOR, 100nF	C84, C1, C2, C3, C4, C5, C6, C35, C36, C37, C38, C34, C33, C32, C31, C30, C29, C28, C27, C26, C25, C24, C23, C22, C21, C20, C19, C18, C17, C16, C15, C14, C13, C12, C11, C10, C9, C8, C7	cap4
3	74LS, 74LS273N	U37, U32, U28	NO20
17	CAPACITOR, 100pF	C75, C74, C73, C72, C71, C70, C69, C68, C67, C66, C65, C64, C63, C62, C61, C60, C59	cap2
1	SOCKETS, DIP28	U24	DIP-28
17	COMPARATOR, LM311N	U3, U5, U7, U35, U33, U31, U29, U27, U25, U23, U21, U19, U17, U15, U9, U11, U13	DIP-8(N08E)
18	RESISTOR, 100kohm	R7, R11, R15, R71, R67, R63, R59, R55, R50, R49, R46, R42, R39, R35, R31, R19, R23, R27	RES0.5
3	POTENTIOMETER, 1K_LIN	R8, R12, R1	LIN_POT
1	RESISTOR, 5.6kohm	R9	RES0.5
18	RESISTOR, 22kohm	R10, R14, R17, R72, R68, R64, R60, R56, R51, R54, R43, R47, R37, R34, R21, R25, R28, R30	RES0.5
1	RESISTOR, 6.8kohm	R13	RES0.5
6	POTENTIOMETER, 10K_LIN	R16, R36, R32, R20, R24, R29	LIN_POT
1	RESISTOR, 9.1kohm	R18	RES0.5
2	OPAMP, LM741CN	U1, U2	DIP-8(N08E)
2	RESISTOR, 20kohm	R4, R6	RES0.5
1	RESISTOR, 10kohm	R3	RES0.5
1	RESISTOR, 24kohm	R2	RES0.5
1	RESISTOR, 1.6ohm	R5	RES0.5
2	RESISTOR, 300kohm	R74, R70	RES0.5
4	POTENTIOMETER, 100K_LIN	R73, R69, R65, R61	LIN_POT
1	RESISTOR, 200kohm	R66	RES0.5
2	RESISTOR, 150kohm	R62, R58	RES0.5
2	POTENTIOMETER, 50K_LIN	R57, R52	LIN_POT
1	RESISTOR, 120kohm	R53	RES0.5
3	POTENTIOMETER, 20K_LIN	R48, R44, R40	LIN_POT
1	RESISTOR, 75kohm	R45	RES0.5
1	RESISTOR, 56kohm	R41	RES0.5
1	RESISTOR, 47kohm	R38	RES0.5
1	RESISTOR, 33kohm	R33	RES0.5
1	RESISTOR, 12kohm	R22	RES0.5
1	RESISTOR, 18kohm	R26	RES0.5

Table 6.5: Bill of Materials for the Modulus 17 PCB.

After the board is stuffed and soldered, power is connected. The component connections are tested and the trim potentiometers are set to provide the correct bias and amplification to the signal and provide the proper reference voltages for the comparators. Now the boards are ready for operation. The Modulus 17 and 8 PCBs are shown in Figures 6.11 and 6.12 respectively. Appendix C gives a detailed narrative of how to manufacture a PCB.

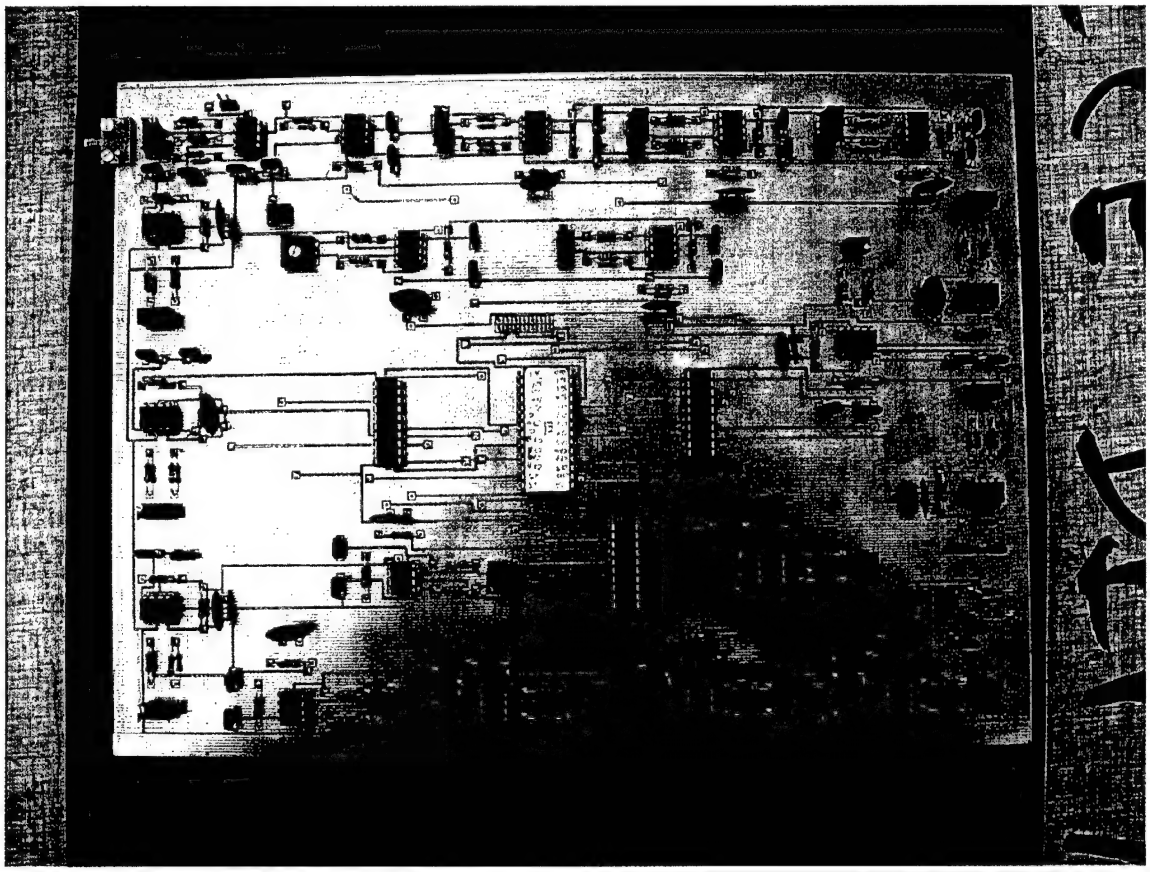


Figure 6.11: Modulus 17 Printed Circuit Board.

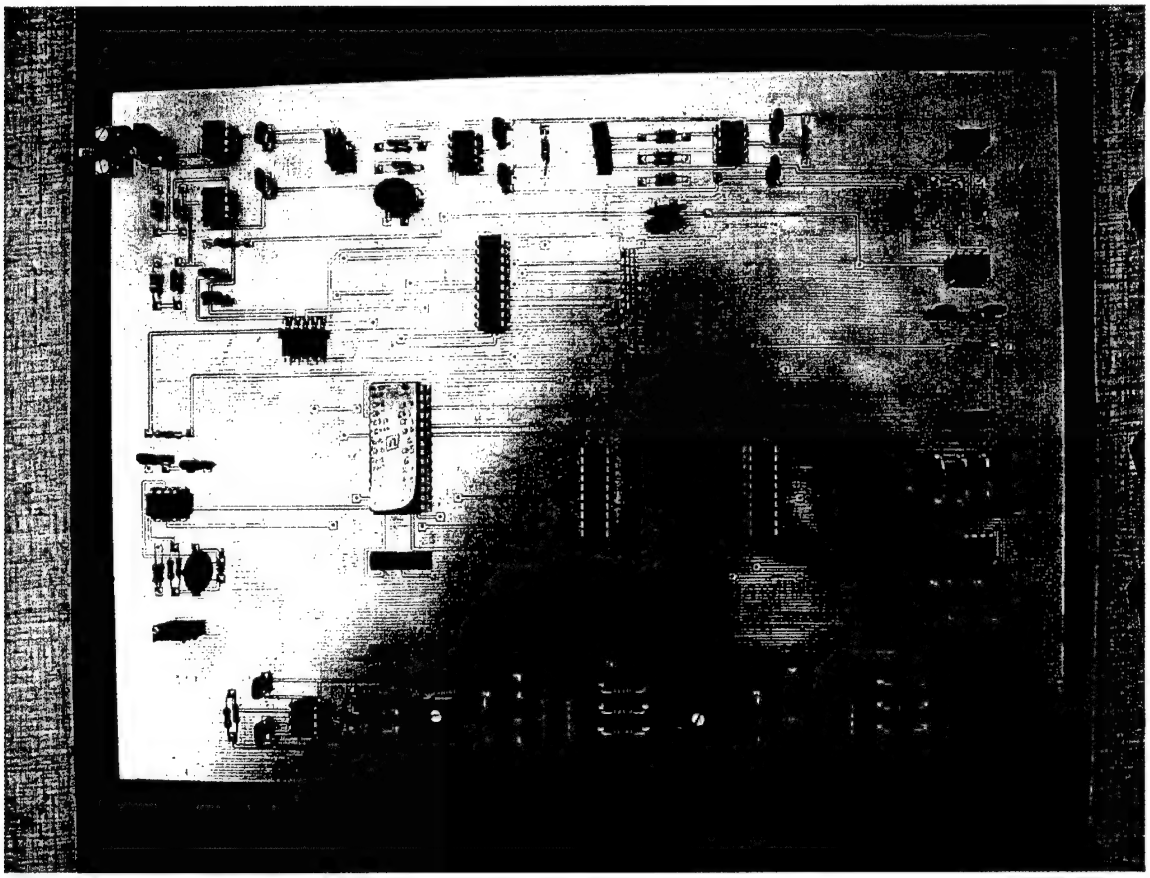


Figure 6.12: Modulus 8 Printed Circuit Board.

VII. TEST AND EVALUATION OF PROTOTYPE HARDWARE

A. TEST PROCEDURES

The prototype antenna was tested in the Naval Postgraduate School Microwave Anechoic Chamber. The major components and communication paths used for testing are shown in Figure 7.1. The test system architecture centerpiece is a Pentium II computer. One of two LABVIEW programs controls the test via a General Purpose Interface Bus (GPIB), the serial port, and a PC-DIO-96 board. The signal generator that produces the radiated signal is located on the same rack as the Hewlett Packard 8510 Vector Network Analyzer (VNA). In the anechoic chamber, a standard gain feed horn transmits the microwave signal generated by the signal generator. The antenna is placed on a pedestal (Figure 7.1) approximately 19 feet from the transmit horn with the microwave circuit mounted on the back of the array ground plane. A servomotor is controlled through the computer serial port and rotates the pedestal, which is capable of steps as small as 0.1°. The microwave circuit continuously analyzes the received signal, and the mixer output is sampled at each resolution step. The mixer output is carried out of the anechoic chamber on RG-59 coaxial cables and either connected to the RSNS processor or two HP 3478A multimeters. The 6-bit output of the RSNS processor is transmitted to the computer via the DIO board. Two HP 3478A digital multimeters measure either the mixer output or the shift and bias circuit output. The computer samples the multimeters through the GPIB bus.

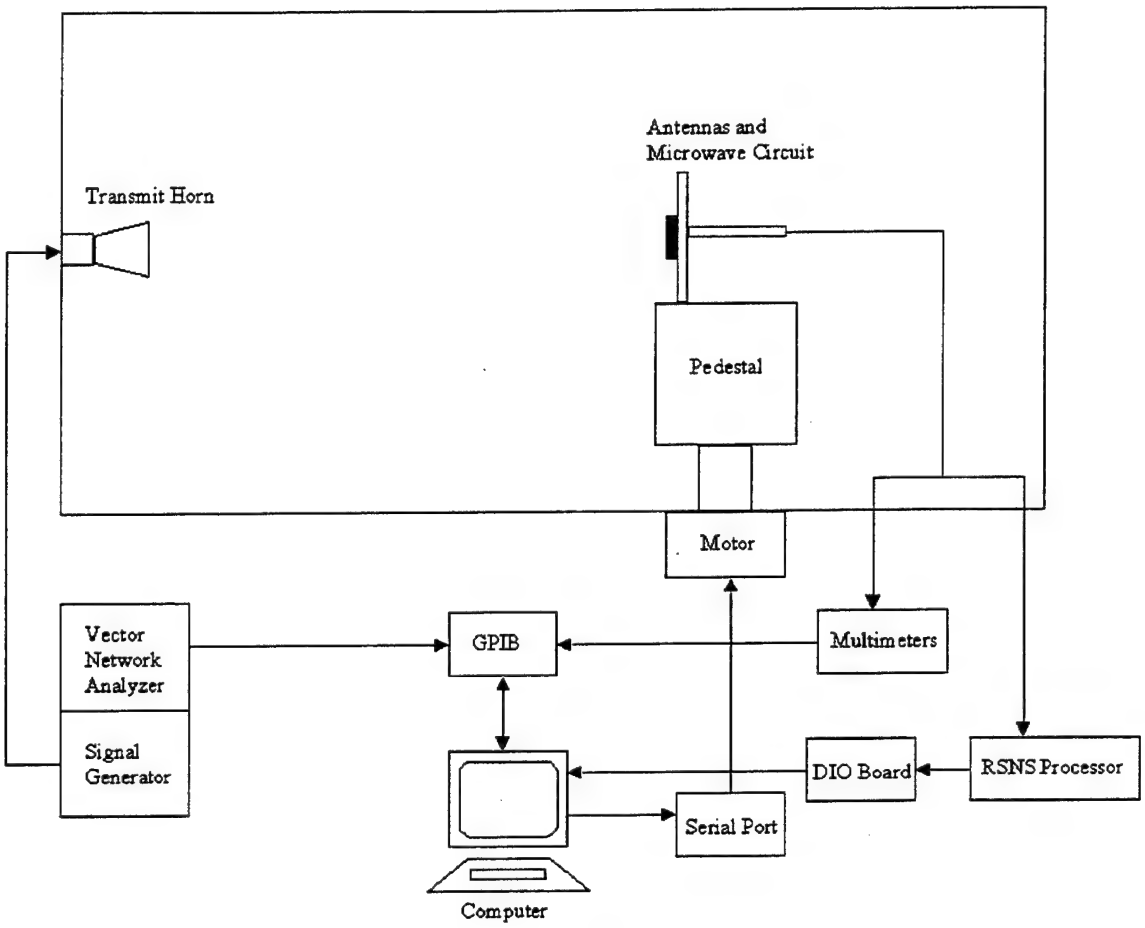


Figure 7.1: Anechoic Chamber Setup.

The data collection and processing technique is illustrated in Figure 7.2. Two types of tests are conducted depending upon the parameter of interest. The first test measures the output of the mixer, and other test measures the output of the shift and bias amplifier and the binary output of the RSNS processor simultaneously. One LABVIEW program is written for each test. When the mixer output is measured, the procedure follows the left path on Figure 7.2. The LABVIEW VI code *DF_mm_R2* samples the multimeters at each servomotor step. The data is saved in column format. The four columns of data are (1) antenna position, (2) channel 1 (modulus 8) multimeter output, (3) antenna position, and (4) channel 2 (modulus 17) multimeter output. The data is analyzed with Microsoft Excel using the "text-tab delimited" format. Manual

normalization of the data is performed. The minimum and maximum values of the phase detector folding waveforms are located. A column is inserted between the second and third data columns. The normalized phase detector output $\eta_{i,norm}$ is calculated using the equations:

$$\overline{\eta} = (\eta_{max} + \eta_{min})/2, \tag{7.1}$$

$$\Delta\eta = (\eta_{max} - \eta_{min})/2, \tag{7.2}$$

$$\eta_{i,norm} = \frac{\eta_i - \overline{\eta}}{\Delta\eta}. \tag{7.3}$$

The normalized output for each channel is placed in the new third column (modulus 8) and the sixth columns (modulus 17). The normalized phase detector output is saved in tab delimited text format with a *".dat"* extension. This is the standard input format for the MATLAB processing programs provided in Appendix A. The measured phase (mixer output) is visually compared to the simulated data using the *rsns8.m* and *rsns17.m* programs. *Phase_error.m* calculates and plots the phase error between a simulated and normalized measured mixer output. The normalized mixer output is passed through a simulation of the digital processing in *Bin_Pop.m.*

The output of the digital circuit follows the path on the right of Figure 7.2. The shift and bias circuit output is connected to the multimeters. The binary output of the digital circuit is connected to the computer via the DIO board. The LABVIEW VI code *DF5_DIO_R2* samples the multimeters and DIO board at each servomotor step. The data is again saved in column format. No normalization is required for this data, but it must be saved in tab delimited text format with a *".dat"* extension. Analysis of the data is performed by one program, *AOA_map.m.* This program plots the folding waveforms and the bin mapping. It converts the bin numbers into an estimated angle of arrival and plots the transfer function.

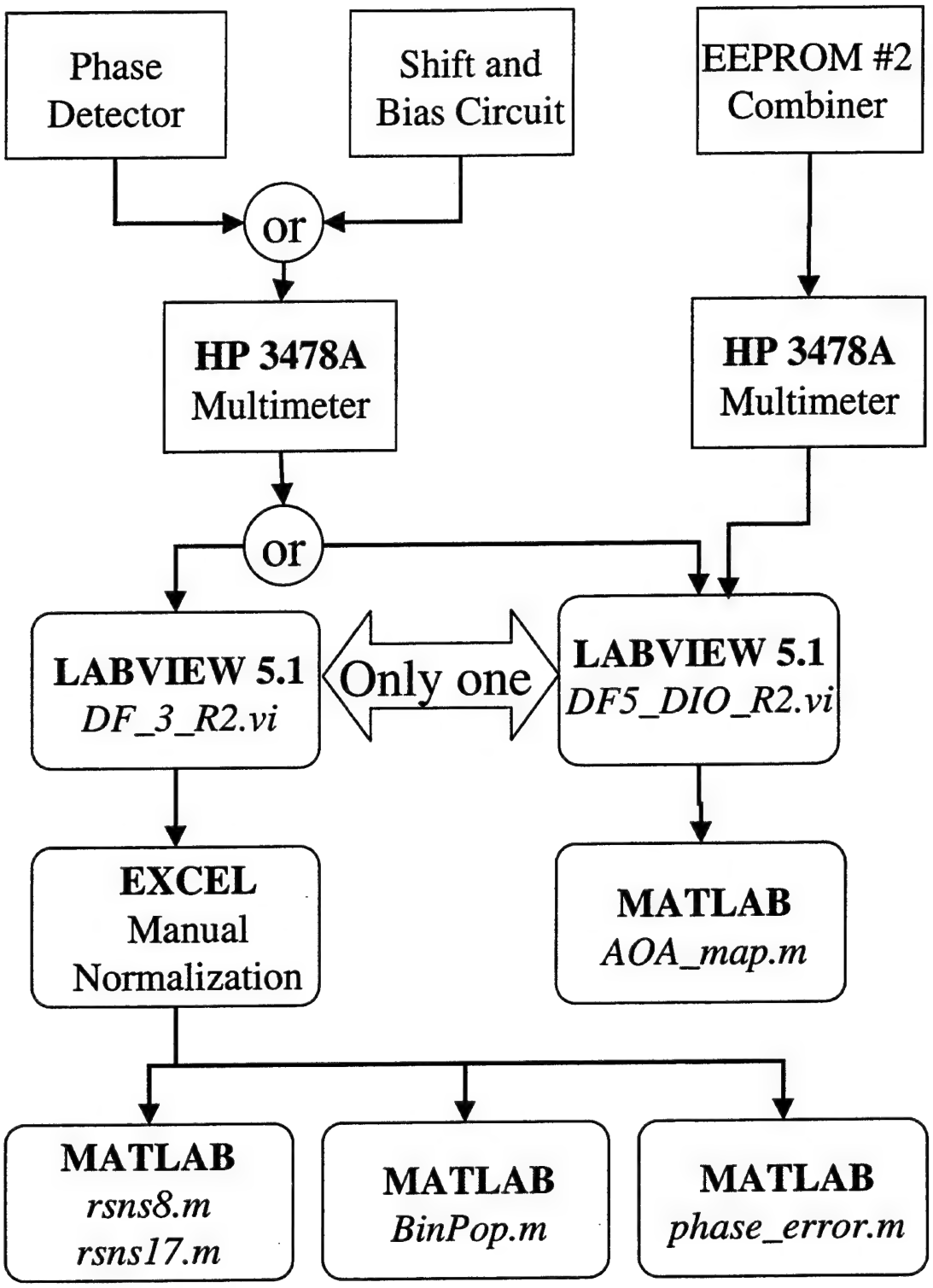


Figure 7.2: Anechoic Chamber Control and Recording Equipment.

B. RESULTS

The RSNS direction finding system was built and tested. The array design is scaled so that the field of view is 60°. The tests were conducted at 8 GHz at a resolution of 0.1°. Data from the mixer outputs, RSNS processor, and level and bias amplifiers were recorded.

The folding waveforms for both moduli are shown in Figures 7.3 and 7.4. The blue curves are the ideal waveform and the red curves are the measured waveforms after they have been normalized. The green curve is the residual error of the measured signal. *BinPop.m* produces the simulated RSNS antenna transfer function using measured mixer outputs. Figure 7.5 shows this transfer function. Once it was shown that the transfer function was mostly continuous, a complete test of the system was conducted. Figure 7.6 is the folding waveforms after the shift and bias amplifiers have processed them. The output of the RSNS processor is recorded through the DIO board and the *AOA_map.m* file produces the angle of arrival estimation of the entire system, shown in Figure 7.7.

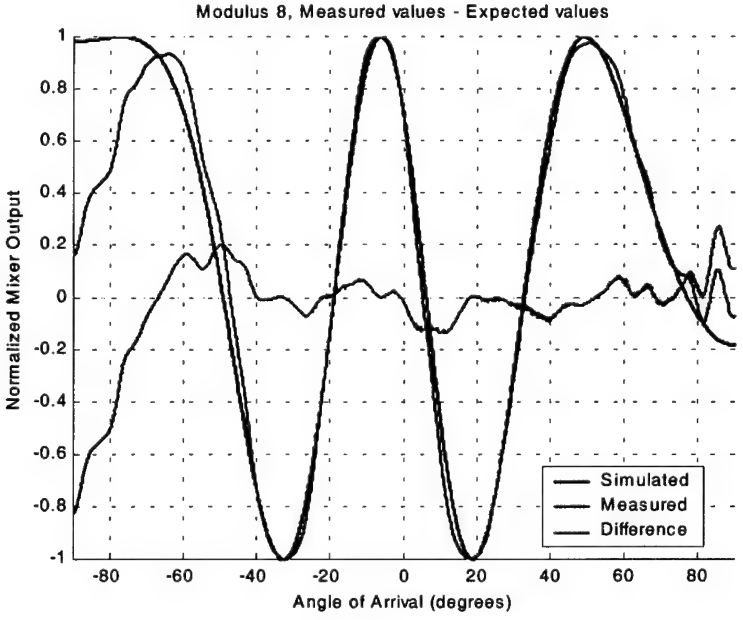


Figure 7.3: Modulus 8 Normalized Mixer Output.

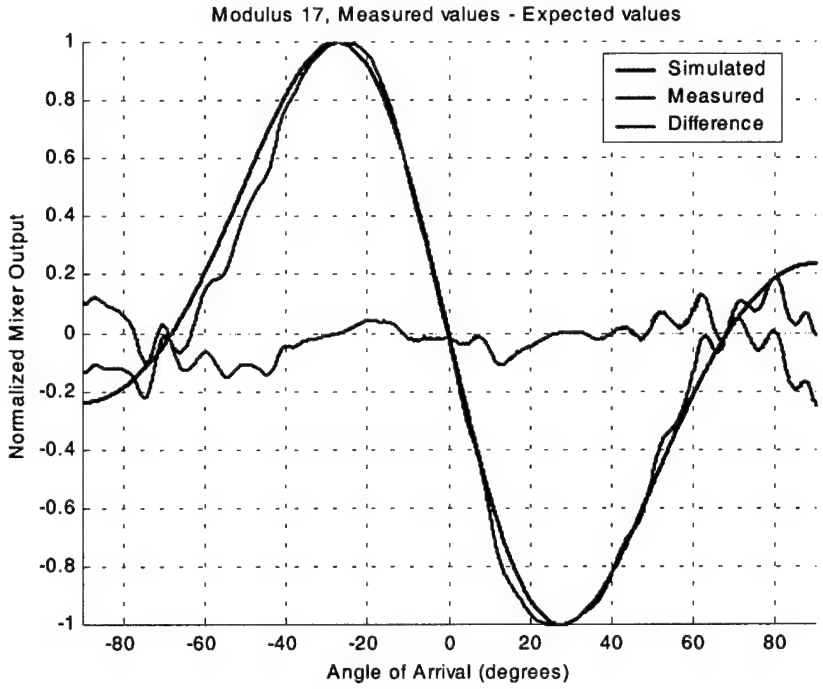


Figure 7.4: Modulus 17 Normalized Mixer Output.

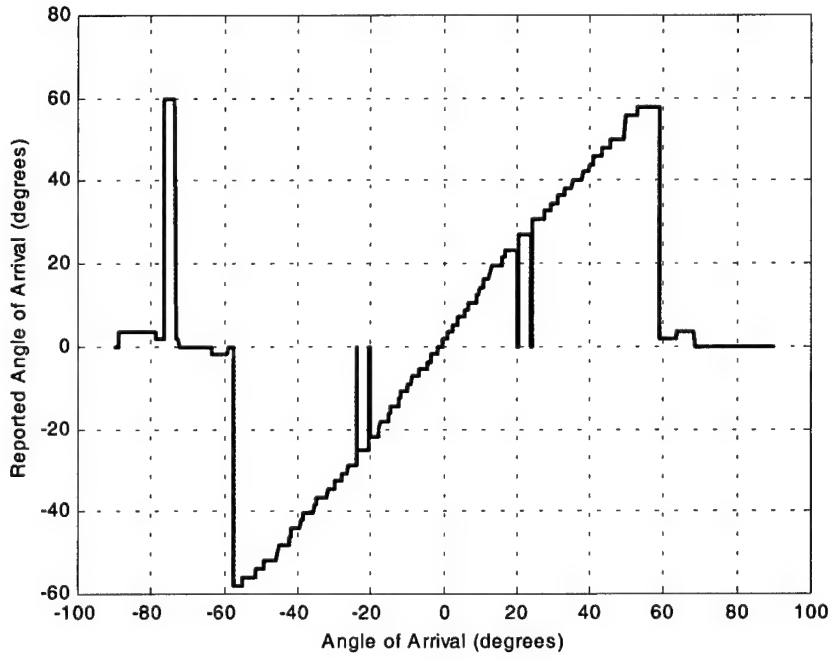


Figure 7.5: Simulated RSNS Antenna Transfer Function Using Measured Mixer Outputs.

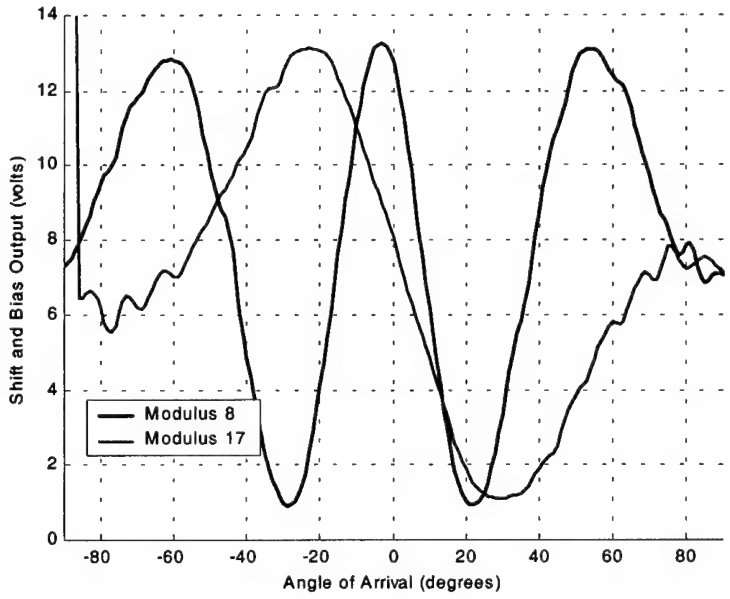


Figure 7.6: Level and Bias Amplifiers Outputs.

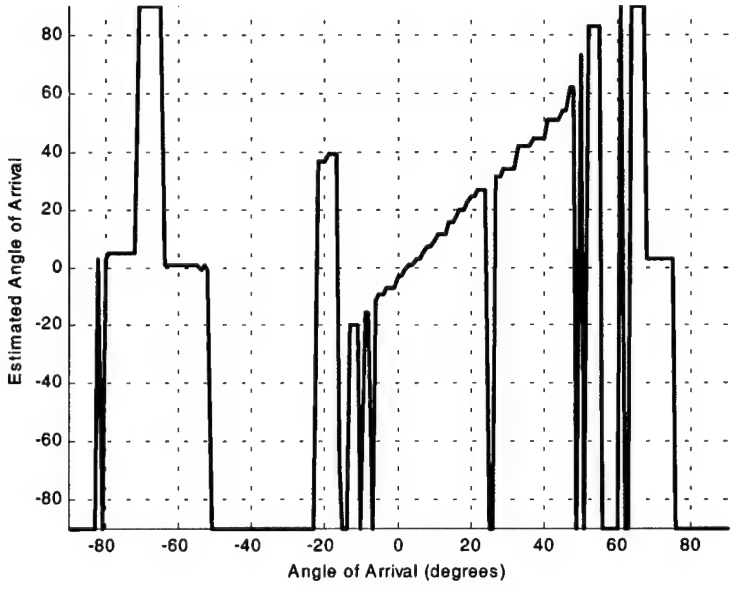


Figure 7.7: RSNS Antenna Transfer Function.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII CONCLUSIONS

A. CONCLUSIONS

This research has introduced the Robust Symmetrical Number System and shown that it can be used to resolve ambiguities in phase sampling interferometry. A direction finding antenna architecture based on the RSNS has many inherent advantages over conventional direction finding techniques including: simple microwave beamforming network that incorporates wideband components, wide instantaneous field of view, and high resolution angle of arrival with a variety of element arrangements having very short baselines. The design equations for comparator thresholds, broadside phase difference, and a scale factor are derived. The simulated transfer function demonstrates that the RSNS antenna will have a wide instantaneous field of view and high resolution.

A second prototype array was built and its performance measured. Several improvements were made to the original RSNS prototype antenna. By adopting the scaled antenna carriage, adding isolators, and completing the ground plane with copper tape between the antenna elements mutual coupling effects are reduced. Mounting the microwave components on a brass plate has reduced errors contributed by vibrations and temperature. Tailor cutting all semi-rigid coaxial lines has reduced the number of connectors required to assemble the microwave circuit and has subsequently reduced the errors inherent in connectors. Matching amplifiers by amplification stage rather than by signal line has reduce relative phase errors between channels as well as better matched the power outputs of both amplification stages. Two printed circuit boards were designed and built for the RSNS processor. The printed circuit boards provide a more stable platform over the original design, which was assembled on breadboards.

The measured transfer function of the new prototype exhibits the basic features of the ideal transfer function. The measured transfer function contains small encoding errors. These errors are attributed to phase errors in the microwave circuit and its components as well as threshold errors inherent in the RSNS processor. Comparing the

transfer functions of the first and second prototypes and the ideal case, the improvements in the system are revealed. Figure 8.1 shows the simulated transfer function of the first prototype RSNS antenna using the measured mixer outputs. Note that the large phase errors present (see Figure 4.7 and 4.8) cause a significant degradation in the antenna performance.

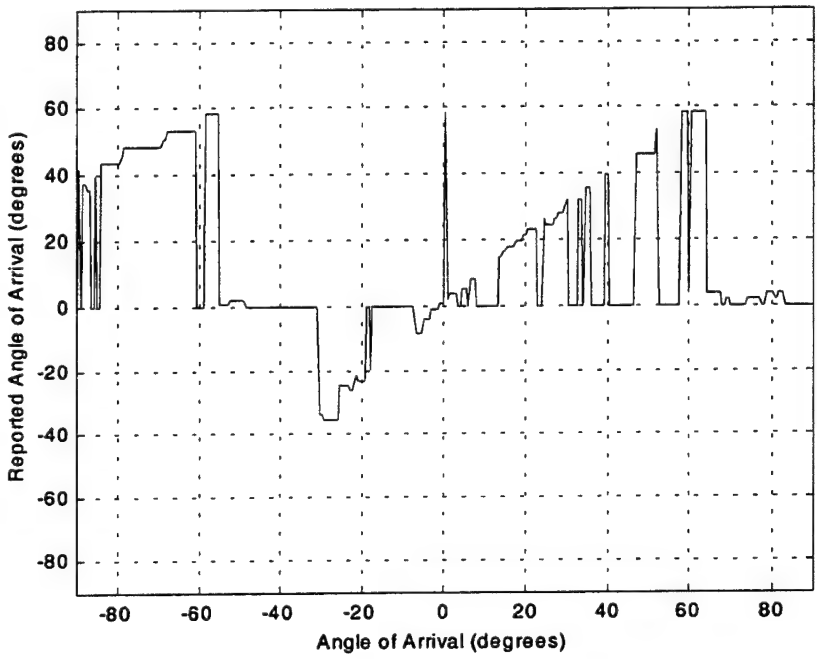


Figure 8.1: Simulated Transfer Function of the RSNS Antenna Using Measured Mixer Outputs (First Prototype) [from 5].

Figure 8.2 shows the simulated transfer function of the second prototype RSNS antenna using the improved mixer output response (see Figure 7.3 and 7.4). Note that this response matches the ideal response (shown in Figure 8.3) much more closely with only a few encoding errors present.

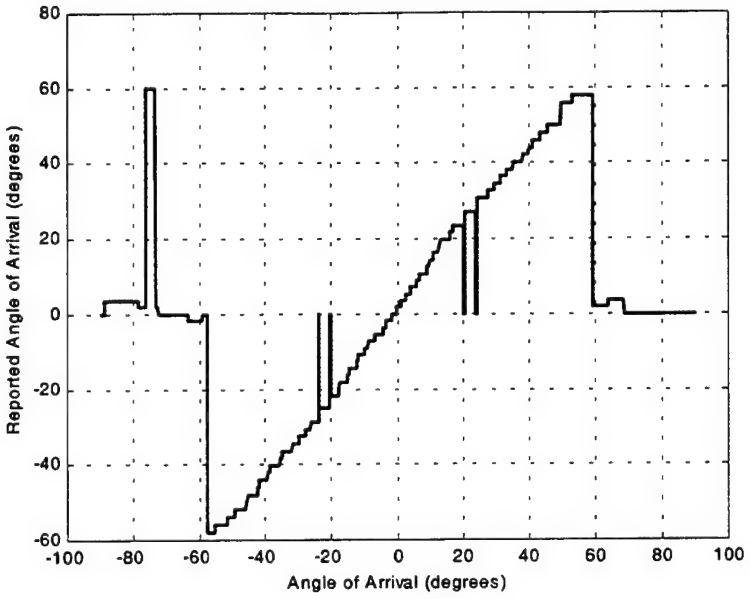


Figure 8.2: Measured Mixer Output Through Simulated RSNS Processor, Second Prototype.

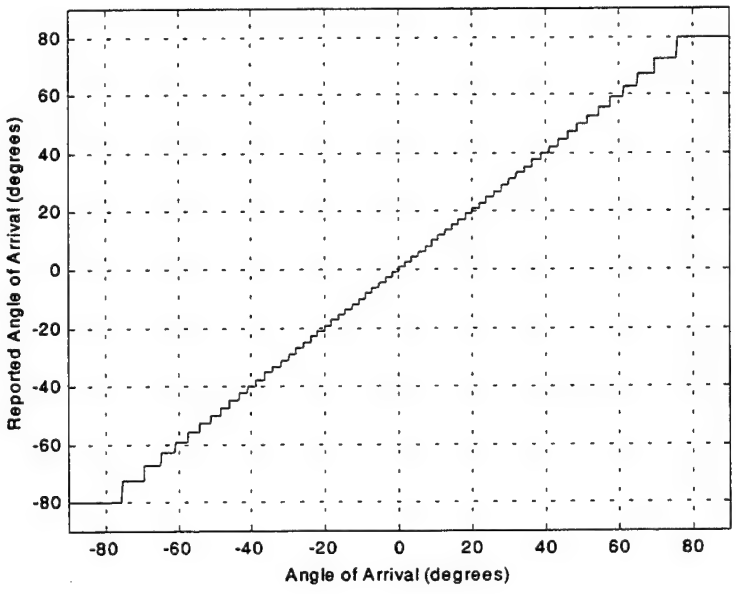


Figure 8.3: Ideal Transfer Function.

The initial results from the RSNS digital processor shown in Figure 8.4 were not expected. The EEPROM programming will output zeros if the inputs do not match a known set of inputs from Table 6.3. This occurs in the regions where the estimated angle of arrival is -90°. Each set of components (i.e. program on EEPROM, latches, comparator thresholds, and bias and level amplifier performance) was individually verified for proper function; however, when the whole system is tested it does not entirely work. Errors in the circuit design may be attributed to the comparators themselves. Data sets from different runs were compared against each other to see where the errors occurred. It was noticed that the errors sometimes occurred at a specific angle and sometimes not. Using the anechoic chamber and testing a single angle verified this. Even though the comparator threshold's accuracy is improved with the Schmitt trigger, errors may still be occurring. To test this, the Labview code needs to be modified so the comparator states, input signal, and threshold values can be monitored. If it is determined that the comparators are not accurate enough, a precision comparator such as the CMP02 or CMP04 could be substituted for the LM311.

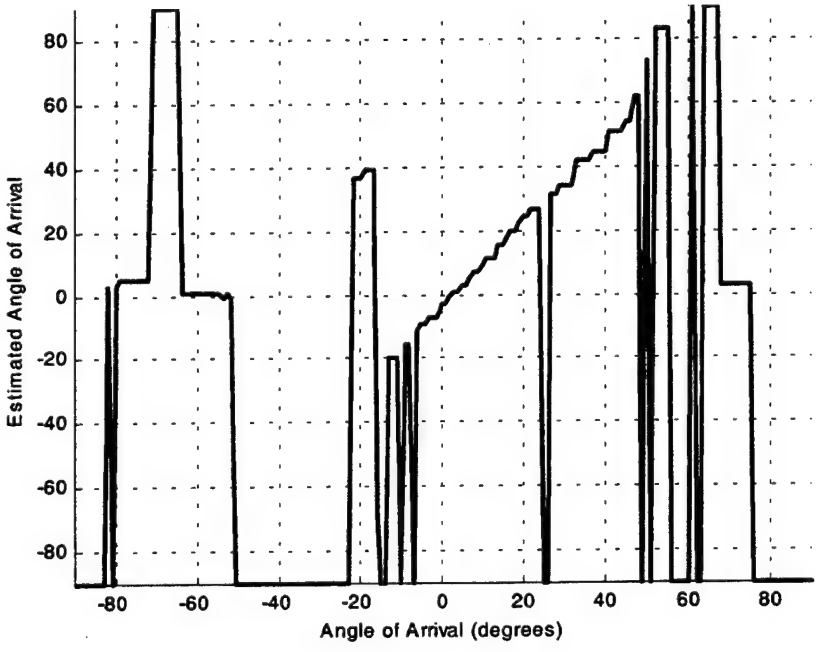


Figure 8.4: Measured Antenna Transfer Function.

B. FUTURE WORK

The complete design of the RSNS direction finding system is an continually evolving process. A number of areas for further research exist. These areas include topics such as frequency agility, increased resolution, and operation in a multi-signal environment.

It is desired to design a new system capable of handling a wide range of frequencies. The angle of incidence of the signal determines the phase difference between the antenna elements. This phase difference is linearly dependant upon the wavelength and therefore the frequency of the incoming signal. A relationship between the angular resolution and the dynamic range of the RSNS can be derived. Research into how to control the frequency effects on the RSNS antenna array is needed.

Another improvement to the design that čan be looked into is increasing the resolution of the system. There are several methods that could be used to accomplish this task. Some ideas are to increase the dynamic range of the RSNS by increasing modulus number in each channel or by increasing the number of channels. Preliminary data suggests that phase errors are more prevalent in higher modulus channels therefore this might not be a desirable method to accomplish increased resolution. Work needs to be conducted to determine the optimal method to accomplish this task.

Finally, in order to be able to field a system in an uncontrolled environment, a multi-signal capability needs to be included into the design. It maybe possible to select individual frequencies by adding an adaptive filter or frequency selector to the front end of the system.

C. CONCLUDING REMARKS

The focus of this thesis research has been to improve the design and performance of a previous RSNS DF array. The design equations and operating parameters have been presented. The overall performance of the microwave circuit has

been improved and the phase errors reduced to a level to where the signal processor con effectively operate. Although the results from the RSNS signal processor are disappointing it has been shown that the RSNS signal processing technique is a viable technique for future DF applications.

APPENDIX A. VHDL CODE

This appendix presents the VHDL code used to simulate both EEPROMs for the system. Each EEPROM needs two modules to function correctly. The first module is the actual code that is used to simulate the component. The second module is a test bench that is used to verify the functionality of the code.

```
--EEPROM #1 Module
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- Multisim VHDL software version: 3.75a
--
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment this if you use
                                  -- Synopsys signed or unsigned types.
entity EEPROM_1 is
    generic (D:time :=10 ns);
    port (
        A0: in bit;
        A1: in bit;
        A2: in bit;
        A3: in bit;
        A4: in bit;
        A5: in bit;
        A6: in bit;
        A7: in bit;
        A8: in bit;
        A9: in bit;
        A10: in bit;
        A11: in bit;
        A12: in bit;
        A13: in bit;
        A14: in bit;
        IO0: out bit;
        IO1: out bit;
        IO2: out bit;
        IO3: out bit
    );
end EEPROM_1;
-- From VHDL Design, J.R. Armstrong, pg. 320
-- Model for combinational logic using CASE statement
architecture EEPROM_1A of EEPROM_1 is
begin
    P1: process(A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0)
    begin
        case A14&A13&A12&A11&A10&A9&A8&A7&A6&A5&A4&A3&A2&A1&A0 is
            when "000000000000000" => IO3 <= '0' after D; IO2 <= '0' after D;
                IO1 <= '0' after D; IO0 <= '0' after D;
            when "000000000000001" => IO3 <= '0' after D; IO2 <= '0' after D;
                IO1 <= '0' after D; IO0 <= '1' after D;
            when "000000000000011" => IO3 <= '0' after D; IO2 <= '0' after D;
                IO1 <= '1' after D; IO0 <= '0' after D;
            when "000000000000111" => IO3 <= '0' after D; IO2 <= '0' after D;
                IO1 <= '1' after D; IO0 <= '1' after D;
            when "000000000001111" => IO3 <= '0' after D; IO2 <= '1' after D;
                IO1 <= '0' after D; IO0 <= '0' after D;
```

```
            when "000000000011111" => IO3 <= '0' after D; IO2 <= '1' after D;
                IO1 <= '0' after D; IO0 <= '1' after D;
            when "000000000111111" => IO3 <= '0' after D; IO2 <= '1' after D;
                IO1 <= '1' after D; IO0 <= '0' after D;
            when "000000001111111" => IO3 <= '0' after D; IO2 <= '1' after D;
                IO1 <= '1' after D; IO0 <= '1' after D;
            when "000000011111111" => IO3 <= '1' after D; IO2 <= '0' after D;
                IO1 <= '0' after D; IO0 <= '0' after D;
            when "000000111111111" => IO3 <= '1' after D; IO2 <= '0' after D;
                IO1 <= '0' after D; IO0 <= '1' after D;
            when "000001111111111" => IO3 <= '1' after D; IO2 <= '0' after D;
                IO1 <= '1' after D; IO0 <= '0' after D;
            when "000011111111111" => IO3 <= '1' after D; IO2 <= '0' after D;
                IO1 <= '1' after D; IO0 <= '1' after D;
            when "000111111111111" => IO3 <= '1' after D; IO2 <= '1' after D;
                IO1 <= '0' after D; IO0 <= '0' after D;
            when "001111111111111" => IO3 <= '1' after D; IO2 <= '1' after D;
                IO1 <= '0' after D; IO0 <= '1' after D;
            when "011111111111111" => IO3 <= '1' after D; IO2 <= '1' after D;
                IO1 <= '1' after D; IO0 <= '0' after D;
            when "111111111111111" => IO3 <= '1' after D; IO2 <= '1' after D;
                IO1 <= '1' after D; IO0 <= '1' after D;
        end case;
    end process P1;
end EEPROM_1A;


------------------------------------------------------
--Test Bench Module for: EEPROM_1
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- Multisim VHDL software version: 3.75a
--
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.all;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component EEPROM_1 is
    port (
        A0: in BIT;
        A1: in BIT;
        A2: in BIT;
        A3: in BIT;
        A4: in BIT;
        A5: in BIT;
        A6: in BIT;
        A7: in BIT;
        A8: in BIT;
        A9: in BIT;
        A10: in BIT;
        A11: in BIT;
        A12: in BIT;
        A13: in BIT;
        A14: in BIT;
        IO0: out BIT;
        IO1: out BIT;
        IO2: out BIT;
        IO3: out BIT
```

```
    );

end component;
constant PERIOD: time := 50 ns;
-- Top level signals go here...
signal A0: BIT;
signal A1: BIT;
signal A2: BIT;
signal A3: BIT;
signal A4: BIT;
signal A5: BIT;
signal A6: BIT;
signal A7: BIT;
signal A8: BIT;
signal A9: BIT;
signal A10: BIT;
signal A11: BIT;
signal A12: BIT;
signal A13: BIT;
signal A14: BIT;
signal IO0: BIT;
signal IO1: BIT;
signal IO2: BIT;
signal IO3: BIT
;
signal done: boolean := false;

begin
    DUT: EEPROM_1 port map (
            A0,
            A1,
            A2,
            A3,
            A4,
            A5,
            A6,
            A7,
            A8,
            A9,
            A10,
            A11,
            A12,
            A13,
            A14,
            IO0,
            IO1,
            IO2,
            IO3
          );


    STIMULUS1: process
    begin

        -- Sequential stimulus goes here...
        --
        -- Sample stimulus...

        A0 <= '0' after 0 ns, '1' after 100 ns;
        A1 <= '0' after 0 ns, '1' after 200 ns;
        A2 <= '0' after 0 ns, '1' after 300 ns;
        A3 <= '0' after 0 ns, '1' after 400 ns;
        A4 <= '0' after 0 ns, '1' after 500 ns;
        A5 <= '0';
        A6 <= '0';
        A7 <= '0';
        A8 <= '0';
        A9 <= '0';
```

```
        A10 <= '0';
        A11 <= '0';
        A12 <= '0';
        A13 <= '0';
        A14 <= '0';

        wait;            -- Suspend simulation
    end process STIMULUS1;

end stimulus;


------------------------------------------------------
--EEPROM #2 Module
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- Multisim VHDL software version: 3.75a
--
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment this if you use
                                  -- Synopsys signed or unsigned types.
entity EEPROM_2 is
    generic (D:time :=10 ns);
    port (
    -- WIZ BEGINPORTS (Entity Wizard command)
        A0: in BIT;
        A1: in BIT;
        A2: in BIT;
        A3: in BIT;
        A4: in BIT;
        A5: in BIT;
        A6: in BIT;
        A7: in BIT;
        A8: in BIT;
        A9: in BIT;
        A10: in BIT;
        A11: in BIT;
        A12: in BIT;
        A13: in BIT;
        IO0: out BIT;
        IO1: out BIT;
        IO2: out BIT;
        IO3: out BIT;
        IO4: out BIT;
        IO5: out BIT
    -- WIZ ENDPORTS (Entity Wizard command)
    );
end EEPROM_2;
-- From VHDL Design, J.R. Armstrong, pg. 320
-- Model for combinational logic using CASE statement
architecture EEPROM_2A of EEPROM_2 is
begin
    P1: process(A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0)
    begin
        case A13&A12&A11&A10&A9&A8&A7&A6&A5&A4&A3&A2&A1&A0 is
            when "00000000000000" => IO5 <= '0' after D; IO4 <= '0' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --0
            when "00111111001010" => IO5 <= '0' after D; IO4 <= '0' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --0
            when "00011111001010" => IO5 <= '0' after D; IO4 <= '0' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '1' after D; --1
```

```
when "00011111001011" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --2
when "00001111001011" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --3
when "00001111001100" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --4
when "00000111001100" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --5
when "00000111001101" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --6
when "00000011001101" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --7
when "00000011001110" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --8
when "00000001001110" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --9
when "00000001001111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --10
when "00000000001111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --11
when "00000000011111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --12
when "00000001011111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --13
when "00000001111111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --14
when "00000011111111" => IO5 <= '0' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --15
when "00000011011111" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --16
when "00000111011111" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --17
when "00000111001111" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --18
when "00001111001111" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --19
when "00001111001110" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --20
when "00011111001110" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --21
when "00011111001101" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --22
when "00111111001101" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --23
when "00111111001100" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --24
```

```
when "01111111001100" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --25
when "01111111001011" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --26
when "11111111001011" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --27
when "11111111001010" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --28
when "01111111001010" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --29
when "01111111001001" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --30
when "00111111001001" => IO5 <= '0' after D; IO4 <= '1' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --31
when "00111111001000" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --32
when "00011111001000" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --33
when "00011111000111" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --34
when "00001111000111" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --35
when "00001111000110" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --36
when "00000111000110" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --37
when "00000111000101" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --38
when "00000011000101" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --39
when "00000011000100" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --40
when "00000001000100" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --41
when "00000001000011" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --42
when "00000000000011" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --43
when "00000000000010" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '0' after D; --44
when "00000001000010" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
    IO0 <= '1' after D; --45
when "00000001000001" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '0' after D; --46
when "00000011000001" => IO5 <= '1' after D; IO4 <= '0' after D;
    IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
    IO0 <= '1' after D; --47
```

```
            when "00000011000000" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --48
            when "00000111000000" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '1' after D; --49
            when "00000111000001" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
                IO0 <= '0' after D; --50
            when "00001111000001" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '0' after D; IO1 <= '1' after D;
                IO0 <= '1' after D; --51
            when "00001111000010" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --52
            when "00011111000010" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '0' after D;
                IO0 <= '1' after D; --53
            when "00011111000011" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
                IO0 <= '0' after D; --54
            when "00111111000011" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '0' after D; IO2 <= '1' after D; IO1 <= '1' after D;
                IO0 <= '1' after D; --55
            when "00111111000100" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --56
            when "01111111000100" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '0' after D;
                IO0 <= '1' after D; --57
            when "01111111000101" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
                IO0 <= '0' after D; --58
            when "11111111000101" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '0' after D; IO1 <= '1' after D;
                IO0 <= '1' after D; --59
            when "11111111000110" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
                IO0 <= '0' after D; --60
            when "01111111000110" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '0' after D;
                IO0 <= '1' after D; --61
            when "01111111000111" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
                IO0 <= '0' after D; --62
            when "00111111000111" => IO5 <= '1' after D; IO4 <= '1' after D;
                IO3 <= '1' after D; IO2 <= '1' after D; IO1 <= '1' after D;
                IO0 <= '1' after D; --63
        end case;
    end process P1;
end EEPROM_2A;


------------------------------------------------------
--Test Bench Module for: EEPROM_2
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- Multisim VHDL software version: 3.75a
--
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.all;
```

```
entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component EEPROM_2 is
    port (
        A0: in BIT;
        A1: in BIT;
        A2: in BIT;
        A3: in BIT;
        A4: in BIT;
        A5: in BIT;
        A6: in BIT;
        A7: in BIT;
        A8: in BIT;
        A9: in BIT;
        A10: in BIT;
        A11: in BIT;
        A12: in BIT;
        A13: in BIT;
        IO0: out BIT;
        IO1: out BIT;
        IO2: out BIT;
        IO3: out BIT;
        IO4: out BIT;
        IO5: out BIT

    );

end component;
constant PERIOD: time := 50 ns;
-- Top level signals go here...
signal A0: BIT;
signal A1: BIT;
signal A2: BIT;
signal A3: BIT;
signal A4: BIT;
signal A5: BIT;
signal A6: BIT;
signal A7: BIT;
signal A8: BIT;
signal A9: BIT;
signal A10: BIT;
signal A11: BIT;
signal A12: BIT;
signal A13: BIT;
signal IO0: BIT;
signal IO1: BIT;
signal IO2: BIT;
signal IO3: BIT;
signal IO4: BIT;
signal IO5: BIT
;
signal done: boolean := false;

begin
    DUT: EEPROM_2 port map (
            A0,
            A1,
            A2,
            A3,
            A4,
            A5,
            A6,
            A7,
            A8,
            A9,
            A10,
```

```
            A11,
            A12,
            A13,
            IO0,
            IO1,
            IO2,
            IO3,
            IO4,
            IO5
          );

    STIMULUS1: process
    begin

        -- Sequential stimulus goes here...
        --
        -- Sample stimulus...

        A0 <= '0', '1' after 100 ns;
        A1 <= '1';
        A2 <= '0', '1' after 100 ns;
        A3 <= '1';
        A4 <= '0';
        A5 <= '0';
        A6 <= '1', '0' after 100 ns;
        A7 <= '1', '0' after 100 ns;
        A8 <= '1', '0' after 100 ns;
        A9 <= '1', '0' after 100 ns;
        A10 <= '1', '0' after 100 ns;
        A11 <= '1', '0' after 100 ns;
        A12 <= '0';
        A13 <= '0';

        wait;              -- Suspend simulation
    end process STIMULUS1;

end stimulus;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB SIMULATION AND DATA EVALUATION CODE

```
%RSNS8 WITH SHIFTING

% LT Nathan York, 15 December 2000
% Generates the simulated folding waveform.  Two element arrays
assumed.
% Can accept normalized mixer output for comparison to simulated
waveform.
% Plots folding waveforms for simulated and measured data.
% m = modulus; d = spacing (m); f = frequency
clear
 N=2;
 f=8.0e9; c=3e8; wavl=c/f;
 k=2*pi/wavl;
 rad=pi/180;
 sf=sqrt(3)/2    % sf is the scale factor


% ffd is forced phase difference to align signal to required zero
crossing
ffd=45;  %original is 45!

% number of moduli in array design and their values
 mm=[8 17]; nm=length(mm);
 disp('moduli are: '),disp(mm)
 % specify a modulus to plot
 m=8;

 % M is Dynamic Range
 M=64;
 % nf is number of folds
 nf=M/(2*m*N);

 % determines required element spacing

 d=nf*wavl/sf/2;

 % start, stop, increment angles
 start=-90; stop=90; inc=0.1;
 N=floor((stop-start)/inc)+1;

 for n=1:N;
   thd(n)=start+(n-1)*inc; thr=thd(n)*rad;
   arg(n)=k*d*sin(thr);
   % mixer output
   mx(n)=cos(arg(n)+(ffd)*(pi/180));
end
```

```
load rsns14.dat
both8=[thd' mx'];
save simu8.dat both8 -ascii


figure(1)
hold
plot(thd,mx,'b');
plot(rsns14(:,1),rsns14(:,3),'r')

hold off
xlabel('Angle of Arrival'),ylabel('Normalized Mixer Output');
grid
xlabel('Angle of Arrival (degrees)')
ylabel('Normalized Mixer Output')
title(['Modulus 8, with ffd = ',num2str(ffd)])
legend('Simulated','Measured',0)
axis([-90,90,-1,1])
orient landscape

temp=rsns14(:,3);
temp1=rot90(temp);
left=temp1-mx;
zero=0;

figure (2)
hold
plot(thd,mx,'b');
plot(rsns14(:,1),rsns14(:,3),'r')
plot(thd,left,'g');
plot(thd,zero,'k');
xlabel('Angle of Arrival (degrees)')
ylabel('Normalized Mixer Output')
legend('Simulated','Measured','Difference',0)
axis([-90,90,-1,1])
grid
title('Modulus 8, Measured values - Expected values')

%RSNS17 WITH SCALING

% LT Nathan York, 15 December 2000
% Generates the simulated folding waveform.  Two element arrays
assumed.
% Can accept normalized mixer output for comparison to simulated
waveform.
% Plots folding waveforms for simulated and measured data.
% m = modulus; d = spacing (m); f = frequency

clear
 N=2;
 f=8e9; c=3e8; wavl=c/f;
 sf=.8746; %sqrt(3)/2
 k=2*pi/wavl;
 rad=pi/180;
```

```
% ffd is forced phase difference to align signal to required zero
crossing
ffd=90  %original is 90!


 m=17;

 % M is Dynamic Range
 M=64;
 % nf is number of folds
 nf=M/(2*m*N);
% calculation of required element spacing
 d=nf*wavl/sf/2;

% start, stop, increment angles
 start=-90; stop=90; inc=0.1;
 N=floor((stop-start)/inc)+1;

 for n=1:N;
   thd(n)=start+(n-1)*inc; thr=thd(n)*rad;
   arg(n)=k*d*sin(thr);
   % mixer output
   mx(n)=cos(arg(n)+(ffd)*(pi/180));


end

load rsns11.dat;
load rsns14.dat;
load PAT9.dat;
load PAT10.dat;
load pat10a.dat

both17=[thd' mx'];
save simu17.dat both17 -ascii;

figure(3)
hold
plot(thd,mx,'b');

% LABVIEW Output has mod 17 mixer output in second column.
%   Normalization technique puts normalized mixer output in third
column.
plot(rsns14(:,4),rsns14(:,6),'r')

hold off
xlabel('Angle of Arrival (degrees)'),ylabel('Normalized Mixer Output');
grid
title(['Modulus 17, with ffd = ',num2str(ffd)])
legend('Simulated','Measured',0)
xlabel('Angle of Arrival (degrees)')
ylabel('Normalized Mixer Output')
axis([-90,90,-1,1])
orient landscape
```

```
temp=rsns14(:,6);
temp1=rot90(temp);
left=temp1-mx;
zero=0;

figure (4)
hold
plot(thd,mx,'b');
plot(rsns14(:,4),rsns14(:,6),'r')
plot(thd,left,'g')
plot(thd,zero,'k')
legend('Simulated','Measured','Difference',0)
grid
xlabel('Angle of Arrival (degrees)')
ylabel('Normalized Mixer Output')
axis([-90,90,-1,1])
title('Modulus 17, Measured values - Expected values')



%BinPop.m

% LT Nathan York 15 December 2000

% step size is 0.1 degrees based on minimum stepper motor resolution
% simulates the entire transfer function
%   1- determines number of active comparators at each step for both
modulus
%   2- determine bin number
%   3- determine angle of arrival

% Also calculates bin population and quantization error.

% Load either the simulated data (simu17.dat, simu8.dat)
% or the experimental data - SEE BELOW (ALSO SEE LINE 155-156)
% Uses predicted thresholds calculated from cosine of equally spaced
angles
%**********************************
format long;
clear;
sum_error_squared=0; Excess_RE=0;
sf=sqrt(3)/2;

%load simu17.dat
%load simu8.dat
%df1=simu17;

load rsns14.dat
df1=[rsns14(:,4) rsns14(:,6)];

%**********************************
point_num=1801;
Sum_AE=0;

%comparators mod 17
```

```
for i=1:point_num

    c_m17(i,[1])=i;
    c_m17(i,[2])=0;
    c_m17(i,[3])=0;
    c_m17(i,[4])=0;
    c_m17(i,[5])=0;
    c_m17(i,[6])=0;
    c_m17(i,[7])=0;
    c_m17(i,[8])=0;
    c_m17(i,[9])=0;
    c_m17(i,[10])=0;
    c_m17(i,[11])=0;
    c_m17(i,[12])=0;
    c_m17(i,[13])=0;
    c_m17(i,[14])=0;
    c_m17(i,[15])=0;
    c_m17(i,[16])=0;
    c_m17(i,[17])=0;
    c_m17(i,[18])=0;

 end

 %SIMULATION THRESHOLDS
   t1  = cos(33*pi/34);
   t2  = cos(31*pi/34);
   t3  = cos(29*pi/34);
   t4  = cos(27*pi/34);
   t5  = cos(25*pi/34);
   t6  = cos(23*pi/34);
   t7  = cos(21*pi/34);
   t8  = cos(19*pi/34);
   t9  =  cos(17*pi/34);
   t10 =  cos(15*pi/34);
   t11  = cos(13*pi/34);
   t12  = cos(11*pi/34);
   t13  = cos(9*pi/34);
   t14  = cos(7*pi/34);
   t15  = cos(5*pi/34);
   t16  = cos(3*pi/34);
   t17  = cos(pi/34);

for i=1:point_num

   % all the data points
   % c1 is the bottom comparator
   % t1 is the threshold voltage for c1 etc.


   if (df1(i,[2]))> t1
      c_m17(i,[2]) = 1;
   else
      c_m17(i,[2]) = 0;
   end
```

```
if (df1(i,[2]))> t2
   c_m17(i,[3]) = 1;
else
   c_m17(i,[3]) = 0;
end

if (df1(i,[2]))> t3
   c_m17(i,[4]) = 1;
else
   c_m17(i,[4]) = 0;
end

if (df1(i,[2]))> t4
   c_m17(i,[5]) = 1;
else
   c_m17(i,[5]) = 0;
end

if (df1(i,[2]))> t5
   c_m17(i,[6]) = 1;
else
   c_m17(i,[6]) = 0;
end

if (df1(i,[2]))> t6
   c_m17(i,[7]) = 1;
else
   c_m17(i,[7]) = 0;
end

if (df1(i,[2]))> t7
   c_m17(i,[8]) = 1;
else
   c_m17(i,[8])= 0;
end

if (df1(i,[2]))> t8
   c_m17(i,[9]) = 1;
else
   c_m17(i,[9]) = 0;
end

if (df1(i,[2]))> t9
   c_m17(i,[10]) = 1;
else
   c_m17(i,[10]) = 0;
end

if (df1(i,[2]))> t10
   c_m17(i,[11]) = 1;
else
   c_m17(i,[11]) = 0;
end
 if (df1(i,[2]))> t11
```

```
      c_m17(i,[12]) = 1;
   else
      c_m17(i,[12]) = 0;
   end

   if (df1(i,[2]))> t12
      c_m17(i,[13]) = 1;
   else
      c_m17(i,[13]) = 0;
   end

   if (df1(i,[2]))> t13
      c_m17(i,[14]) = 1;
   else
      c_m17(i,[14]) = 0;
   end

   if (df1(i,[2]))> t14
      c_m17(i,[15]) = 1;
   else
      c_m17(i,[15]) = 0;
   end

   if (df1(i,[2]))> t15
      c_m17(i,[16]) = 1;
   else
      c_m17(i,[16]) = 0;
   end

   if (df1(i,[2]))> t16
      c_m17(i,[17]) = 1;
   else
      c_m17(i,[17]) = 0;
   end

   if (df1(i,[2]))> t17
      c_m17(i,[18]) = 1;
   else
      c_m17(i,[18]) = 0;
   end


end
      %encoding the thermometer code into a decimal number.
      %rsns_m17 contains the number of comparators on at each time
      %step

for j=1:point_num

          A=c_m17(j,[2]);
          B=c_m17(j,[3]);
          C=c_m17(j,[4]);
          D=c_m17(j,[5]);
          E=c_m17(j,[6]);
          F=c_m17(j,[7]);
```

```
            G=c_m17(j,[8]);
            H=c_m17(j,[9]);
            I=c_m17(j,[10]);
            J=c_m17(j,[11]);
            K=c_m17(j,[12]);
            L=c_m17(j,[13]);
            M=c_m17(j,[14]);
            N=c_m17(j,[15]);
            O=c_m17(j,[16]);
            P=c_m17(j,[17]);
            Q=c_m17(j,[18]);

            Z=A+B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q;

            rsns_m17(j)=Z;

end

xarray=[-90:180/1800:90];
figure(6)
plot(xarray,rsns_m17(:)),grid
xlabel('Angle of Arrival (degrees)')
ylabel('Number of Comparators')
% title('Number of Comparators on each time step for mod 17')
axis([-90 90 -1 17])

%*********************************
clear df1

%df1=simu8;

df1=[rsns14(:,1) rsns14(:,3)];
%*********************************

%comparators mod 8

for i=1:point_num

    c_m8(i,[1])=i;
    c_m8(i,[2])=0;
    c_m8(i,[3])=0;
    c_m8(i,[4])=0;
    c_m8(i,[5])=0;
    c_m8(i,[6])=0;
    c_m8(i,[7])=0;
    c_m8(i,[8])=0;
    c_m8(i,[9])=0;

end

t1 =  cos(15*pi/16);
t2 =  cos(13*pi/16);
t3 =  cos(11*pi/16);
t4 =  cos(9*pi/16);
t5 =  cos(7*pi/16);
```

```
t6 =  cos(5*pi/16);
t7 =  cos(3*pi/16);
t8 =  cos(pi/16);

for i=1:point_num

   % all the data points
   % c1 is the bottom comparator
   % t1 is the threshold voltage for c1 etc

   if (df1(i,[2]))> t1
      c_m8(i,[2]) = 1;
   else
      c_m8(i,[2]) = 0;
   end

   if (df1(i,[2]))> t2
      c_m8(i,[3]) = 1;
   else
      c_m8(i,[3]) = 0;
   end

   if (df1(i,[2]))> t3
      c_m8(i,[4]) = 1;
   else
      c_m8(i,[4]) = 0;
   end

   if (df1(i,[2]))> t4
      c_m8(i,[5]) = 1;
   else
      c_m8(i,[5]) = 0;
   end

   if (df1(i,[2]))> t5
      c_m8(i,[6]) = 1;
   else
      c_m8(i,[6]) = 0;
   end

   if (df1(i,[2]))> t6
      c_m8(i,[7]) = 1;
   else
      c_m8(i,[7]) = 0;
   end

   if (df1(i,[2]))> t7
      c_m8(i,[8]) = 1;
   else
      c_m8(i,[8]) = 0;
   end

   if (df1(i,[2]))> t8
      c_m8(i,[9]) = 1;
   else
```

```
      c_m8(i,[9]) = 0;
   end

end

      %encoding the thermometer code into a decimal number.
      %rsns_m8 contains the number of comparators on at each time
      %step

for j=1:point_num

          A=c_m8(j,[2]);
          B=c_m8(j,[3]);
          C=c_m8(j,[4]);
          D=c_m8(j,[5]);
          E=c_m8(j,[6]);
          F=c_m8(j,[7]);
          G=c_m8(j,[8]);
          H=c_m8(j,[9]);
          Z=A+B+C+D+E+F+G+H;

          rsns_m8(j)=Z;

end

figure(7)
plot(xarray,rsns_m8(:),'r'),grid
xlabel('Angle of Arrival (degrees)')
ylabel('Number of Comparators')
% title('Number of Comparators on each time step for m8')
axis([-90 90 -1 8])

figure(8)
hold on
plot(xarray,rsns_m17(:),'b')
plot(xarray,rsns_m8(:),'r')
hold off
xlabel('Angle of Arrival (degrees)')
ylabel('Number of Comparators')
legend('Mod 8','Mod 17',0)
%title('Number of Comparators on each time step for mod 17 & mod 8')
axis([-90 90 0 17])
orient landscape


% integers for trans_funct file
bin=-1; bin_ctr=1; start_th=-90; end_th=0; sum_error_squared=0;

% Bin mapping for thermometer code.  doa is the bin number.
for s=1:point_num

      if (rsns_m17(s)==10 & rsns_m8(s)==6)
         doa(s) = 0;

      elseif (rsns_m17(s)==10 & rsns_m8(s)==5)
```

```
   doa(s) = 1;

elseif (rsns_m17(s)==11 & rsns_m8(s)==5)
   doa(s) = 2;

elseif (rsns_m17(s)==11 & rsns_m8(s)==4)
   doa(s) = 3;

elseif (rsns_m17(s)==12 & rsns_m8(s)==4)
   doa(s) =4;

elseif (rsns_m17(s)==12 & rsns_m8(s)==3)
   doa(s) = 5;

elseif (rsns_m17(s)==13 & rsns_m8(s)==3)
   doa(s) = 6;

elseif (rsns_m17(s)==13 & rsns_m8(s)==2)
   doa(s) = 7;

elseif (rsns_m17(s)==14 & rsns_m8(s)==2)
   doa(s) = 8;

elseif (rsns_m17(s)==14 & rsns_m8(s)==1)
   doa(s) = 9;

elseif (rsns_m17(s)==15 & rsns_m8(s)==1)
   doa(s) = 10;

elseif (rsns_m17(s)==15 & rsns_m8(s)==0)
   doa(s) = 11;

elseif (rsns_m17(s)==16 & rsns_m8(s)==0)
   doa(s) = 12;

elseif (rsns_m17(s)==16 & rsns_m8(s)==1)
   doa(s) = 13;

elseif (rsns_m17(s)==17 & rsns_m8(s)==1)
   doa(s) = 14;

elseif (rsns_m17(s)==17 & rsns_m8(s)==2)
   doa(s) = 15;

elseif (rsns_m17(s)==16 & rsns_m8(s)==2)
   doa(s) = 16;

elseif (rsns_m17(s)==16 & rsns_m8(s)==3)
   doa(s) = 17;

elseif (rsns_m17(s)==15 & rsns_m8(s)==3)
   doa(s) = 18;

elseif (rsns_m17(s)==15 & rsns_m8(s)==4)
   doa(s) = 19;
```

```
elseif (rsns_m17(s)==14 & rsns_m8(s)==4)
   doa(s) = 20;

elseif (rsns_m17(s)==14 & rsns_m8(s)==5)
   doa(s) = 21;

elseif (rsns_m17(s)==13 & rsns_m8(s)==5)
   doa(s) = 22;

elseif (rsns_m17(s)==13 & rsns_m8(s)==6)
   doa(s) = 23;

elseif (rsns_m17(s)==12 & rsns_m8(s)==6)
   doa(s) = 24;

elseif (rsns_m17(s)==12 & rsns_m8(s)==7)
   doa(s) = 25;

elseif (rsns_m17(s)==11 & rsns_m8(s)==7)
   doa(s) = 26;

elseif (rsns_m17(s)==11 & rsns_m8(s)==8)
   doa(s) = 27;

elseif (rsns_m17(s)==10 & rsns_m8(s)==8)
   doa(s) = 28;

elseif (rsns_m17(s)==10 & rsns_m8(s)==7)
   doa(s) = 29;

elseif (rsns_m17(s)==9 & rsns_m8(s)==7)
   doa(s) = 30;

elseif (rsns_m17(s)==9 & rsns_m8(s)==6)
   doa(s) = 31;

elseif (rsns_m17(s)==8 & rsns_m8(s)==6)
   doa(s) = 32;

elseif (rsns_m17(s)==8 & rsns_m8(s)==5)
   doa(s) = 33;

elseif (rsns_m17(s)==7 & rsns_m8(s)==5)
   doa(s) = 34;

elseif (rsns_m17(s)==7 & rsns_m8(s)==4)
   doa(s) = 35;

elseif (rsns_m17(s)==6 & rsns_m8(s)==4)
   doa(s) = 36;

elseif (rsns_m17(s)==6 & rsns_m8(s)==3)
   doa(s) = 37;
```

```
elseif (rsns_m17(s)==5 & rsns_m8(s)==3)
   doa(s) = 38;

elseif (rsns_m17(s)==5 & rsns_m8(s)==2)
   doa(s) = 39;

elseif (rsns_m17(s)==4 & rsns_m8(s)==2)
   doa(s) = 40;

elseif (rsns_m17(s)==4 & rsns_m8(s)==1)
   doa(s) = 41;

elseif (rsns_m17(s)==3 & rsns_m8(s)==1)
   doa(s) = 42;

elseif (rsns_m17(s)==3 & rsns_m8(s)==0)
   doa(s) = 43;

elseif (rsns_m17(s)==2 & rsns_m8(s)==0)
   doa(s) = 44;

elseif (rsns_m17(s)==2 & rsns_m8(s)==1)
   doa(s) = 45;

elseif (rsns_m17(s)==1 & rsns_m8(s)==1)
   doa(s) = 46;

elseif (rsns_m17(s)==1 & rsns_m8(s)==2)
   doa(s) = 47;

elseif (rsns_m17(s)==0 & rsns_m8(s)==2)
   doa(s) = 48;

elseif (rsns_m17(s)==0 & rsns_m8(s)==3)
   doa(s) = 49;

elseif (rsns_m17(s)==1 & rsns_m8(s)==3)
   doa(s) = 50;

elseif (rsns_m17(s)==1 & rsns_m8(s)==4)
   doa(s) = 51;

elseif (rsns_m17(s)==2 & rsns_m8(s)==4)
   doa(s) = 52;

elseif (rsns_m17(s)==2 & rsns_m8(s)==5)
   doa(s) = 53;

elseif (rsns_m17(s)==3 & rsns_m8(s)==5)
   doa(s) = 54;

elseif (rsns_m17(s)==3 & rsns_m8(s)==6)
   doa(s) = 55;

elseif (rsns_m17(s)==4 & rsns_m8(s)==6)
```

```
         doa(s) = 56;

      elseif (rsns_m17(s)==4 & rsns_m8(s)==7)
         doa(s) = 57;

      elseif (rsns_m17(s)==5 & rsns_m8(s)==7)
         doa(s) = 58;

      elseif (rsns_m17(s)==5 & rsns_m8(s)==8)
         doa(s) = 59;

      elseif (rsns_m17(s)==6 & rsns_m8(s)==8)
         doa(s) = 60;

      elseif (rsns_m17(s)==6 & rsns_m8(s)==7)
         doa(s) = 61;

      elseif (rsns_m17(s)==7 & rsns_m8(s)==7)
         doa(s) = 62;

      elseif (rsns_m17(s)==7 & rsns_m8(s)==6)
         doa(s) = 63;

      else
         doa(s) = 100;  % if the vector does not map, 100 is the error
code

   end


   % Uses the formula asin[(#-34)/64] to find bin center
   gamma=doa(s)+1;
   bin_center(s)=2*asin((gamma-32)*sf/64)*180/3.1415;

   % if
   if doa(s)==100
      bin_center(s)=0;     % if the vector did not map, reports as
broadside
   end
   angle_error(s)=bin_center(s)-(s-900)/10;

   %Calculation of RMS Reporting Error
   %if statement eliminates excessive errors and subs with mean error.
   if abs(angle_error(s)) <= 15
      Sum_AE = Sum_AE + angle_error(s)*angle_error(s);
      Excess_RE=Excess_RE+1;
   end
end

z=0:63;

for l=1:64
   bin_count(l)=0;
end
```

```
for lb =1:1801
   if doa(lb)==100
      bc=1;
   else
      bc=doa(lb)+1;
   end
   bin_count(bc)=bin_count(bc)+1;
end


figure(9)
  hold
  plot(z,bin_count,'b'), axis([0,63,0,60]),grid
  hold off
  xlabel('Bin Number')
  ylabel('Bin Population')


figure(10)
   plot(xarray,doa,'b'),axis([-90,90,0,65])
   xlabel('Angle of Arrival (degrees)')
   ylabel('Bin Number')
   grid

figure(11)
   plot(xarray,bin_center,'b'),grid
   xlabel('Angle of Arrival (degrees)')
   ylabel('Reported Angle of Arrival (degrees)')


figure(12)
   plot(xarray,angle_error,'b'),grid
   axis([-60,60,-10,10]);
   ylabel('Reporting Error (degrees)')
   xlabel('Angle of Arrival (degrees)')

% end of program.


% phase_error.m

% Determines the phase error between a predicted and measured waveform
% 2 channel input
% Written by Dave Wickersham

clear all;

load rsns14.dat
theta=[rsns14(:,1)];

dv8=[rsns14(:,3)];
dv17=[rsns14(:,6)];
md=length(dv8);

% constants and design variables
```

```
N=2;
f=8e9; c=3e8; wavl=c/f;
sf=2/sqrt(3);
k=2*pi/wavl;
M=64;
nf=M/(2*N);% nf is number of folds (not really, no modulus in this
equation)
dw=nf/2;   % new design for mixer ouputs

FirstMin=0;FirstMax=0;SecMin=0;SecMax=0;Max17=0;Min17=0;

rad=pi/180;

thet_r=theta*rad;

ffd8=45;      %45
ffd17=90;    %90

e8_sq=0; e17_sq=0; cnt8=0; cnt17=0;
d8=dw*wavl*sf/8;
d17=dw*wavl*sf/17;

%subroutine to find local max and min
% works because I know about where the max and min will fall
% used to get the right quadrant for the arccos
%Mod 8
for i=(md-1)/6:(md-1)/2
   if dv8(i)<FirstMin
      FirstMin=dv8(i);
      Index1Min=i;
   end
end

for i=(md-1)/3:(md-1)/2
   if dv8(i)>FirstMax
      FirstMax=dv8(i);
      Index1Max=i;
   end
end

for i=(md-1)/2:2*(md-1)/3
  if dv8(i)<SecMin
      SecMin=dv8(i);
      Index2Min=i;
   end
end

for i=2*(md-1)/3:8*(md-1)/9
   if dv8(i)>SecMax
      SecMax=dv8(i);
      Index2Max=i;
   end
end

%Mod 17
```

```
for i=(md-1)/2:5*(md-1)/6
   if dv17(i)<Min17
      Min17=dv17(i);
      Index17Min=i;
   end
end

for i=(md-1)/6:(md-1)/2
   if dv17(i)>Max17
      Max17=dv17(i);
      Index17Max=i;
   end
end



for n=1:md
   delt8(n)=k*d8*sin(thet_r(n))+(ffd8)*rad;   %delt8 is the simulated
total angle in rad
   unwrap(delt8(n));
   %while delt8(n)>pi
   %   delt8(n)=delt8(n)-2*pi;
   %end
   %while delt8(n)<-pi
   %   delt8(n)=delt8(n)+2*pi;
   %end

   dm8(n)=acos(dv8(n));    %dm8 is measured data's total angle in rad
%  unwrap(dm8(n);
   if (n>=Index1Min)&(n<=Index1Max)
      dm8(n)=-dm8(n);
   end
   if (n>=Index2Min)&(n<=Index2Max)
      dm8(n)=-dm8(n);
   end


   phi8(n)=(delt8(n)-dm8(n));
%  unwrap phi8(n);
   while phi8(n)>pi
      phi8(n)=phi8(n)-2*pi;
   end
   while phi8(n)<-pi
      phi8(n)=phi8(n)+2*pi;
   end



  delt17(n)=k*d17*sin(thet_r(n))+(ffd17)*rad;   %delt17 is the
simulated total angle in rad
   %   unwrap(delt17(n));
   while delt17(n)>pi
      delt17(n)=delt17(n)-2*pi;
   end
   while delt17(n)<-pi
```

```
      delt17(n)=delt17(n)+2*pi;
   end

   dm17(n)=acos(dv17(n));     %dm17 is measured data's total angle in
rad

   if n>=Index17Min
      dm17(n)=-dm17(n);
   end

   if n<=Index17Max
      dm17(n)=-dm17(n);
   end


   phi17(n)=(delt17(n)-dm17(n));
   while phi17(n)>pi
      phi17(n)=phi17(n)-2*pi;
   end
   while phi17(n)<-pi
      phi17(n)=phi17(n)+2*pi;
   end



end

phi8=phi8/rad;
phi17=phi17/rad;


for i=1:md                                    %phi contributes to RMS error
only if it is real
   e8_sq = e8_sq + (real(phi8(i))*real(phi8(i)));
   e17_sq = e17_sq+ (real(phi17(i))*real(phi17(i)));
end

rms8=sqrt(e8_sq)/(md-1);              %calculate RMS phase error for each
channel
rms17=sqrt(e17_sq)/(md-1);

disp(['RMS Phase Error of Mod 8  is ',num2str(rms8),' '])
disp(['RMS Phase Error of Mod 17 is ',num2str(rms17),' '])


figure(5)
hold
%plot(p12(:,4),p12(:,8),'--r')
plot(theta,phi8,'b');
plot(theta,phi17,'r');
axis([-90,90,-90,90])
hold off
legend('Mod 8','Mod17',0)
xlabel('Angle of Arrival (degrees)'),ylabel('Phase Error (degrees)');
grid
```

```
%title(['Phase Error between Simulated and Measured Data, Pattern 32'])
orient landscape
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. PRINTED CIRCUIT BOARD CONSTRUCTION

The following is a list of steps that were used to create the printed circuit boards for this thesis. These steps can streamline the construction process for someone who is not familiar with the process. The instructions are only valid when using the Electronic Workbench Multisim software package and the PCB milling machine operated by the Physics Department at NPS.

1. Start Multisim. Create and test the circuit you would like to build. Be sure to include all power sources and grounds.
2. Select the **Transfer to Ultiboard** option in the Transfer pull down menu. This will automatically start the Ultiboard program and import a parts list and netlist that is used by Ultiboard.
3. Setup the Ultiboard workspace for your design. To do this you will have to select a design class, board size, number of layers, and trace size. Use the defaults for all the other options.
4. You will notice a the outlines for all the components at the top of the screen. You will have to manually place each of the components within the board outline. Each component will have a line ending in a circle emanating from the part's center. This is a placement, aid which will help you determine the optimum placement for each component. The shorter the lead the better. As you place components in the board outline you will also notice yellow lines connecting the pins of the selected component and others. These are your pin connections. This can help you determine the appropriate orientation of the components.
5. Ensure that all the power connections are on the board. These are the only parts that are not included by the Multisim program. You will need to manually add the connectors to the board and manually add each of the pins into the correct netlist.
6. Once all the parts are in their postions save the design file and transfer the layout to the Ultiroute program. This program will automatically route all the traces for your

board. In most cases, the first attempt may no complete all the traces. If this happens, close the program and go back to the Ultiboard program. You will need to reconfigure the location of the parts for a second run. If this happens, ensure that you check your grid size. Ultiroute sets the gridsize to 12.5 mils by default even if the design file uses a different grid setting. Make sure that you change the grid setting back to its original, otherwise the design will not transfer to Ultiboard for a second run. You need to continue this process until the board is 100% routed.

7. After the board is routed, transfer it back to the Ultiboard program. Use this program to check for continuity, power grids, pin connections and ground lines. One tip to make the construction easier is making all pin connections on the bottom layer. This will prevent you from having to solder underneath a component.
8. Once this is complete, it is time to generate a Gerber file. This is a standard computer aided manufacturing (CAM) file type for all PCB milling machines. Under the File menu select **Post Processing**. Here you will be prompted to the output device or file type. Select the Gerber RS-274-D (2.3) file type in the Photoplotter menu. Next double-click on each of the four layers to bring up the Plot Setting Parameters. Unselect the Extended Borders and Board Outline option. These option will cause the software used to run the milling machine to crash. The click on the **Go!** button. The Gerber files *.G0 and *.G1 have now been created. The click on the **Drill Targets** button and add that information to the processed file.
9. Copy the .G0, .G1 and .REP files to a 3 and a half inch disk and bring the disk down to the milling machine.
10. The milling machine is running off a 486 computer using DOS 6.0. First read the .REP file and copy down all the apertures that are in the design file. These will have to be manually added to the Gerber file.
11. Rename the .G0 and .G1 files to TOP.GBR and BOT.GBR respectively.
12. Copy these two files to the C:\CAM directory.
13. Start the Gerber editor by typing CAM.EXE.
14. Start a new design file and load the two .GRB files in it.

15. The next step is to edit the aperture list. The program will prompt you to do this. Just add the apertures from the .REP file to the list.
16. Now it is time to edit the .GBR files themselves. Evaluate the design that was transferred from the Ultiboard program and make sure that all the traces and pads are present. At this point you will be able to modify the trace and pad sizes. I suggest that you use 20 mil traces and 80 by 80 mil square pads. These are the easiest to solder to.
17. Once you are satisfied with the .GBR files, save your work and exit the program.
18. Copy TOP.GBR and BOT.GBR to C:\PBOARD\INPUT.
19. Rename TOP.GBR and BOT.GBR to L1.GBR and L2.GBR respectively.
20. Using the EDIT program in DOS, open the L1 and L2 Gerber files and delete the G01 from the third line and save.
21. Type P1.BAT to start the outline converter. This program will convert the Gerber files to machine language that will control the milling machine. The outline converter may take up to two hours to convert a large design.
22. Once the outline converter is done, type P3.BAT. This copies the output files to the CAM directory.
23. Run the CAM program again. This time add L1OUT, L2OUT and D1OUT files to the original design file. You will be able to check the new Gerber files against the originals to make sure the outline converter did its job properly. Most likely, the outline converter will miss a trace or pad. You will need to manually complete the trace in this program. If you need to change these files, make sure you go back and edit them to remove the G01 from the third line. Then copy the new files back to C:\PBOARD\OUTPUT.
24. You are now ready to start milling. Type P2.BAT to start the milling program. The program will have prompts at the bottom of the screen to help you along.
25. Place board material in milling machine.
26. Start by setting the OPS Mode to Drill.
27. Select the layer D1OUT.

28. Go to the Manual option to change out the drill bit and set drill depth. Use a 32 mil drill bit, and set the drill depth so that the bit completely goes through the board material. Once this is set, quit and return to the main menu.
29. Turn on the drill and vacuum, and close the machine cover. The select Start. Follow the prompts to start drilling all the holes.
30. Once the holes are drilled, it is time to cut the traces. To do this set the OPS Mode to mill and select the L1OUT layer.
31. Go to manual to change the drill bit to a milling bit. Set the depth of cut so that the channel is 10 mils wide. A couple of test cuts in the manual mode will be required to do this.
32. Go to Start and follow the prompts to start milling the traces.
33. Once the top layer is complete you need to cut the bottom layer. Follow the same procedure. The L2OUT layer needs to be selected as well as Mirror is ON selected. The bit and depth should be ok so go straight to Start and begin milling the bottom layer.
34. Now that the board has been cut you need to check the board for milling defects. Start by sanding both sides of the board with a very fine grit sandpaper.
35. Check for continuity and grounds. Do this using a multimeter. If a ground is detected, clean the channels with an exacto-knife. There is probably a small filing that was not completely removed by the drill bit that is causing the ground.
36. When the board has been completely checked, it is time to stuff and solder it. Start by making the vias (through board connections). If there are not many vias, small pieces of wire can be soldered to both sides of the board. Another method is using small aluminum rivets. Put the rivets in the drill holes and turn the board upside down on a completely flat working surface. Use a punch to secure the rivet to the bottom of the board. Make sure that there is a good connection between the rivet and both sides of the board; otherwise you will have continuity problems. If you have trouble getting good contact with the rivets, a drop of solder on each side will correct the problem.

37. Now that the vias are complete, finish stuffing the board and soldering the pins and components into place. One hint: solder a few pins at a time and then check for grounds. It is very easy to accidentally jump the channel with the solder causing a ground. If check after a few pins it will be easier to find the mistake and correct it.
38. The board is now ready to be used.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. M.D. Zoltowski and K.T. Wong, "ESPRIT-Based 2-D Direction Finding with a Sparse Uniform Array of Electromagnetic Vector Sensors," *IEEE Transactions on Signal Processing*, Vol. 48, No. 8, pp. 2195-2204, August 2000

2. N. Dowlut and A. Manikas, "A Polynomial Rooting Approach to Super-Resolution Array Design," *IEEE Transactions on Signal Processing*, Vol. 48, No. 6, pp. 1559-1569, June 2000

3. Y.W. Wu, S. Rhodes, and E.H. Satorius, "Direction of Arrival Estimation Via Extended Phase Interferometry," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 1, pp. 375-382, January 1995

4. Thomas Hatziathanasiou, "Optimum Symmetrical Number System Phase Sampled Direction Finding Antenna Architectures," Naval Postgraduate School Master's Thesis, June 1998.

5. D. Wickersham, "Application of the Robust Symmetrical Number System to High Resolution Direction Finding Interferometry," Naval Postgraduate School Master's Thesis, March 2000.

6. P.E. Pace, *Advanced Techniques for Digital Receivers,* Artech House Inc., Norwood MA, 2000

7. P.E. Pace, D. Styer, "High resolution encoding process for an integrated optical analog-to-digital converter," *Optical Engineering*, Vol. 33, pp. 2638-2645, August 1994.

8. D. Jenn, P. Pace, T. Hatziathanasiou, "High Resolution Wideband Direction Finding Arrays Based On Optimum Symmetrical Number System Encoding," *Electronics Letters*, Vol 34, No. 11, May 28, 1998, pp. 1062-1063.

9. P. E. Pace, D. Styer and I. A. Akin, "A folding ADC preprocessing architecture employing a robust symmetrical number system with Gray-code properties," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 47, No. 5, May 2000.

10. .J. Helszajn, *Passive and Active Microwave Circuits*, John Wiley & Sons, New York, 1976, pp. 215-230

11. P. Horowitz, and W. Hill, *The Art of Electronics*, Cambridge University Press, New York, 1989, pp.565-671

12. S.S. Sedra, and K.C Smith, *Microelectronic Circuits*, Oxford University Press, New York, 1998

13. C.H., Roth, *Digital Systems Design Using VHDL*, PWS Publishing Company, Boston, 1998, pp. 85-100

14. J.R Armstrong, and F.G. Gray, *VHDL Design Representation and Synthesis*, Prentice Hall PTR, Upper Saddle River, NJ, 2000, pp 315-323

INITIAL DISTRIBUTION LIST

No of Copies

1. Defense Technical Information Center. 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, Ca 93943-5101

3. Chairman, Code PH 2
   Department of Physics
   Naval Postgraduate School
   Monterey, CA, 93943-5121

4. Professor Phillip E. Pace, Code EC/PC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA, 93943-5121

5. Professor D. Scott Davis, Code PH/DV 1
   Physics Department
   Naval Postgraduate School
   Monterey, CA, 93943-5121

6. Professor David C. Jenn, Code EC/JN 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA, 93943-5121

7. Director, Center for Reconnaissance Research 2
   (Attention: Professor John Powers, Code EC/PO)
   Naval Postgraduate School
   Monterey, CA, 93943-5121

8. Engineering and Technology Curricular Office, Code 34 1
   Naval Postgraduate School
   Monterey, CA, 93943-5121

9. Professor David Styer 1
   16 Brandywine Dr.
   Cincinnati, OH, 45246-3809

10. Lieutenant David Wickersham 1
    1304 Mallory Ct.
    Norfolk VA 23507

11. Lieutenant Nathan S. York 1
    32 Grove St
    Belmont, MA, 02478